Politecnico di Torino

Politecnico di Torino

Master's Degree in Electronics for Industrial Applications
Academic Year 2025–2026

Power-Efficiency Optimizations for RISC-V Based UAV Controllers

Implementation and Evaluation on the NEORV32 Core

Supervisors:
Prof. Maurizio Martina
Christian Fibich, MSc

Candidate:
Francesco Pasquale de Ceglia
Student ID Italy: 344616
Student ID Vienna: io25x501

Estimated period: February 2026 – June 2026

Abstract

This thesis investigates hardware clock gating as a power-efficiency technique for FPGA-based RISC-V embedded controllers, using the open-source NEORV32 processor as the target architecture. The motivation is the reduction of unnecessary dynamic activity during non-computational intervals, which are common in interrupt-driven embedded systems and UAV-like control workloads.

The proposed approach combines the standard RISC-V `WFI` instruction with a BUFGCE-based gated clock domain implemented on Xilinx Artix-7 FPGAs. At hardware level, the NEORV32 system was modified to introduce a dedicated CPU clock, `clk_cpu`, whose enable is controlled by the processor sleep state and by the required wake-up sources, including timer and interrupt signals. At software level, the CoreMark benchmark was instrumented to alternate active computation chunks with timer-driven sleep intervals, enabling a controlled comparison between continuous execution, WFI-only behavior, a permanently enabled clock-path control case (`CE = 1`), and the fully clock-gated implementation.

The experimental validation was carried out on two Artix-7 based platforms. The CW305 board was used for oscilloscope-based power-trace analysis through the shunt measurement path, while the Arty A7-100T platform was used for an integrated XADC-based acquisition flow. On the Arty platform, a phase-aware hardware FIFO was implemented so that each XADC sample is stored together with the phase in which it was acquired, avoiding software-side phase mislabeling during UART readout.

The results show that software sleep and hardware clock gating have distinct effects. Executing `WFI` reduces useful processor activity, but a significant amount of clock-driven switching remains when the clock is still propagated. On the Arty A7-100T platform, the fully clock-gated configuration reduced the low-activity VCCINT power by approximately 18 mW with respect to the `CE = 1` control case, corresponding to a reduction of about 26%. The active-to-sleep power difference in the clock-gated configuration remained close to 35 mW across the tested CoreMark iteration counts.

These results demonstrate that BUFGCE-based clock gating is an effective and measurable low-power mechanism for FPGA-based RISC-V controllers. Although the experimental workload is based on CoreMark rather than a complete UAV control application, the results validate the proposed technique under controlled active/sleep conditions and provide a foundation for future evaluation with more realistic sensor-driven workloads.

Funding Acknowledgment

This work was conducted in the context of the project "Stadt Wien Kompetenzteam für Drohnentechnik in der Fachhochschulausbildung" (project number MA23 35-02), financed by the Department MA23 for Economic Affairs, Labour and Statistics of the City of Vienna.

Contents

Chapter 1

Introduction

Nowadays, microprocessors represent the fundamental core of the global digital landscape, having evolved from the first prototypes of 1969 into the highly versatile 32- and 64-bit architectures that drive today's embedded systems. Within this historical progression, the emergence of the RISC-V Instruction Set Architecture (ISA) has redefined the industry through its open-source nature and modular reconfigurability, allowing hardware designers to adapt processor implementations to specific application requirements.

This thesis presents an experimental study of power-efficiency optimizations for RISC-V-based embedded controllers, with a particular focus on FPGA implementations and their relevance to battery-powered systems such as Unmanned Aerial Vehicles (UAVs). In these applications, energy autonomy is a critical constraint: every reduction in unnecessary power consumption can contribute to longer operating time, lower thermal stress, and more efficient use of the available energy budget.

Modern UAV controllers and similar embedded systems do not always execute useful computation continuously. Instead, they often alternate between active processing windows and waiting intervals, for example while waiting for timers, sensor data, communication events, or external interrupts. During these intervals, the processor may be idle from the software point of view, but the clock network and the sequential logic can still remain active. This leads to unnecessary dynamic power consumption, especially in FPGA-based soft processors where the clock distribution network and clocked resources contribute significantly to switching activity.

The primary objective of this work is to evaluate whether a standard RISC-V sleep mechanism can be combined with FPGA-safe hardware clock gating to reduce this unnecessary activity. The target architecture is the open-source NEORV32 processor, implemented on Xilinx Artix-7 FPGA platforms. The proposed approach modifies the NEORV32 hardware to introduce a dedicated CPU clock domain driven through a BUFGCE primitive, while the software uses the RISC-V `WFI` (Wait-For-Interrupt) instruction to enter controlled sleep intervals.

The choice of `WFI` is deliberate. Rather than introducing a custom software register or a non-standard power-control instruction, the gating mechanism is linked to a standard architectural feature of the RISC-V ISA. This makes the approach more repeatable and conceptually portable to other RISC-V implementations. At the hardware level, the BUFGCE primitive is used because clock gating in FPGA fabric must not be implemented through generic LUT-based logic, which could introduce glitches or unsafe clock pulses. The use of a dedicated FPGA clocking resource allows the CPU clock to be suppressed during valid sleep phases while preserving the wake-up infrastructure required to resume

execution.

To evaluate the proposed method under controlled conditions, the CoreMark benchmark is used as the software workload. CoreMark does not represent a complete UAV flight-control application; rather, it provides a deterministic and repeatable workload that can be divided into active computation chunks and timer-driven sleep intervals. This structure makes it possible to isolate the effect of software sleep, the effect of the modified clock path, and the additional contribution of actual hardware clock suppression.

The experimental validation follows a multi-layered approach. First, the NEORV32 hardware and software are modified to support active/sleep execution, GPIO-based triggering, and timer-based wake-up. Then, the design is implemented and analyzed in the FPGA design flow. Finally, physical measurements are performed on two Artix-7 based platforms. The CW305 board is used for oscilloscope-based observation of the shunt-related activity signal, while the Arty A7-100T platform is used for an integrated XADC-based acquisition flow.

The Arty A7-100T campaign required an additional refinement of the measurement architecture. In the first acquisition approach, the XADC samples were labelled by the software only when they were read back through UART. This was not sufficiently robust, because the FIFO could contain samples acquired during a different phase. For this reason, the final measurement flow stores each XADC sample together with the hardware phase in which it was acquired. A capture-enable mechanism was also introduced to avoid filling the FIFO during UART dumping. This phase-aware acquisition allows active, sleep, and reference intervals to be compared more reliably.

The results obtained in this work show that software sleep and hardware clock gating are distinct effects. Executing `WFI` reduces useful processor activity, but when the clock is still physically propagated a significant amount of clock-driven switching remains. The fully clock-gated configuration provides the largest reduction. On the Arty A7-100T platform, the clock-gated configuration reduces the low-activity VCCINT power by approximately 18 mW with respect to the `CE = 1` control case, corresponding to a reduction of about 26%. The active-to-sleep difference in the clock-gated case remains close to 35 mW across the tested CoreMark iteration counts.

In summary, the path taken by this research moves from the theoretical fundamentals of CMOS dynamic power dissipation to the practical implementation of an FPGA-safe clock-gating strategy for a RISC-V soft processor. The work combines hardware modification, software instrumentation, FPGA implementation, and physical measurements in order to evaluate the real contribution of BUFGCE-based CPU clock suppression. Although the experimental workload is based on CoreMark rather than on a complete UAV control loop, the results provide a concrete proof of concept for applying standard RISC-V sleep semantics to power-aware FPGA-based embedded controllers.

The remainder of this thesis is organized as follows. Chapter 2 introduces the required background on RISC-V, the NEORV32 processor, CMOS power consumption, and clock gating techniques. Chapter 3 reviews related work on low-power RISC-V processors, FPGA clock gating, and physical power validation. Chapter 4 presents the proposed approach and methodology. Chapter 5 describes the hardware platforms and software tools used in the experimental work. Chapter 6 defines the experimental setup and measurement methodology. Chapter 7 presents and discusses the results obtained on the CW305 and Arty A7-100T platforms. Finally, Chapter 8 concludes the thesis and outlines possible future developments.

Chapter 2

Background

2.1 RISC-V General Architecture

The RISC-V Instruction Set Architecture (ISA), pronounced "risk-five", is a free and open standard based on the established principles of Reduced Instruction Set Computer (RISC) design. Unlike proprietary architectures such as x86 or ARM, RISC-V is managed by a non-profit foundation, allowing for royalty-free hardware implementations.

Although the standard was formally transferred to RISC-V International in 2015, its development began in 2010 at the University of California, Berkeley. Architecturally, it is defined as a load-store architecture; this paradigm dictates that computational operations cannot be performed directly on data stored in the main memory. Instead, data must be explicitly loaded into a register file, processed, and subsequently stored back if necessary.

The RISC-V specifications support multiple address space variants, most notably 32-bit (RV32) and 64-bit (RV64) versions, with a 128-bit variant (RV128) defined for future high-scale applications. Instructions are encoded into a fixed 32-bit length and are categorized into six primary base formats:

- **R-type**: register-to-register operations.
- **I-type**: immediate and load operations.
- **S-type**: store operations.
- **B-type**: conditional branching.
- **U-type**: long immediates.
- **J-type**: unconditional jumps.

A defining characteristic of RISC-V is its modularity. The ISA is composed of a mandatory base integer set (e.g., RV32I) which can be augmented with optional extensions to tailor the processor for specific tasks. Common extensions include:

- **M-extension**: for integer multiplication and division.
- **F, D, and Q extensions**: for single, double, and quad-precision floating-point operations.
- **C-extension**: for compressed 16-bit instructions to improve code density.

- **Zicsr extension**: for Control and Status Register (CSR) access.

This modularity is particularly advantageous in the context of Unmanned Aerial Vehicles (UAVs) and low-power embedded systems, as it allows designers to strip away unnecessary logic, thereby reducing both silicon area and power dissipation.

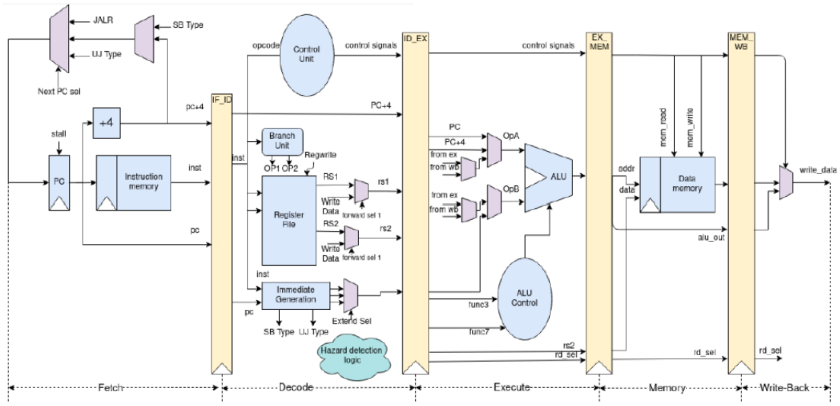


Fig. 3. The RISC-V ISA compliant RV32IM 5-Stage fully pipelined datapath designed from scratch with Chisel

Figure 2.1: Classic 5-stage RISC-V Pipeline architecture.

The baseline RISC-V architecture typically adopts a classic five-stage pipeline to achieve a balance between throughput and implementation complexity. The stages are defined as follows:

1. **Instruction Fetch (IF):** During this stage, the instruction is retrieved from memory using the address currently held in the Program Counter (PC). The PC is incremented by 4 for sequential execution or updated with a target address in the event of a jump or a taken branch.

2. **Instruction Decode (ID):** The 32-bit instruction is decomposed to extract the opcode and register specifiers. The source registers ($rs1, rs2$) are read from the Register File (RF), and any immediate values or memory offsets are sign-extended. This stage also identifies the destination register index (rd) for the final write-back.

3. **Execute (EX):** The Arithmetic Logic Unit (ALU) performs the operation specified by the instruction on the identified operands (OpA and OpB). Control signals derived from the *funct3* and *funct7* instruction fields determine the specific ALU function. In complex **PIU** (Process Interface Unit) implementations, forwarding logic is integrated here to resolve data hazards by bypassing the memory or write-back stages.

4. **Memory Access (MEM):** This stage is active only for load and store instructions. The ALU-computed address is used to access the data memory. For load operations, the data is retrieved; for store operations, the content of $rs2$ is written to memory. If no memory access is required, the stage is bypassed, and the ALU result is passed directly to the final stage.

5. **Write-Back (WB):** The final outcome of the instruction, whether it be an ALU result or data loaded from memory, is written into the destination register rd within the Register File. This completes the instruction life cycle, making the result available for subsequent operations.

While this 5-stage model represents the foundational pipeline, advanced implementations—including the NEORV32 core used in this work—may introduce architectural variations or specialized hardware primitives to optimize specific metrics such as timing closure on FPGA or dynamic power consumption.

2.2 The NEORV32 Processor and SoC Architecture

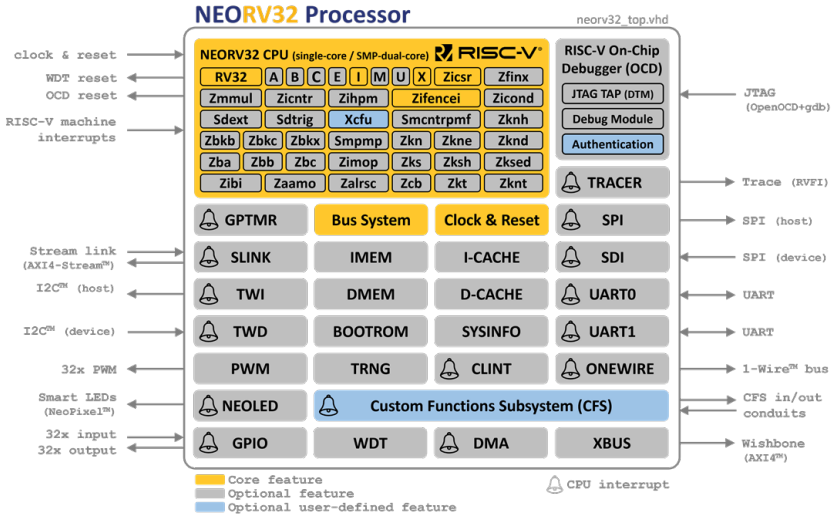


Figure 2.2: NEORV32 processor.

The NEORV32 is a size-optimized, modular, and highly customizable System-on-Chip (SoC) platform based on the RISC-V architecture, developed by Stephan Nolting. Implemented in platform-independent *VHDL*, the system is specifically designed for FPGA deployment, allowing designers to instantiate only the necessary modules to meet specific application requirements.

This hardware modularity is a key feature that aligns with the energy-efficiency goals of this work, as it enables the removal of unused peripherals to minimize silicon area and static power dissipation.

At the heart of the platform is the **NEORV32 CPU**, a Von Neumann architecture that combines a multi-cycle execution scheme with an efficient pipeline. The system is supported by several internal memory components:

- **Instruction Memory (IMEM)** and **Data Memory (DMEM)**: These are implemented as internal FPGA resources and can be configured via generic parameters. While their size is adjustable, it is essential to update the software linker script to reflect any modifications.
- **Bootloader ROM (BOOTLDROM)**: This contains the firmware responsible for system initialization. It facilitates interaction with the CPU for application uploading, manages non-volatile flash memory operations, and initiates the autonomous boot sequence.

For the experimental setup of this project, a specific subset of peripherals has been implemented:

- **UART Channels**: Two channels (*UART0* and *UART1*) sharing an identical hardware structure—supporting an 8-bit frame, one stop bit, and no parity. The baud rate is dynamically configurable through a dedicated clock prescaler.
- **GPIO Module**: Although the specification supports up to 64 pins, this implementation is restricted to 8 pins, serving as the primary interface for general-purpose signaling and control.

Beyond the specific modules used in this research, the NEORV32 platform provides an extensive ecosystem:

- **High-Performance Interconnects**: A *Wishbone* interface and a **Custom Function Subsystem (CFS)** for hardware accelerators.
- **Memory Optimizations**: Optional instruction and data caches (*iCache* and *dCache*).
- **Communication Interfaces**: Support for *SPI*, *TWI* (I2C), *ONEWIRE*, and *SLINK*.
- **Timers and PWM**: *Watchdog Timer* (WDT), *Machine System Timer* (MTIME), and up to 12 independent **Pulse Width Modulation (PWM)** channels.
- **Security and Reliability**: Hardware **True Random Number Generator (TRNG)** and **Cyclic Redundancy Check (CRC)** module.
- **Debugging**: A J-TAG accessible **On-Chip Debugger**, essential for the verification of low-level power states.

The synergy between the base RISC-V ISA and this comprehensive SoC environment provides a robust framework for implementing and validating the RTL-level power optimizations explored in this work.

2.3 NEORV32 CPU Internal Microarchitecture

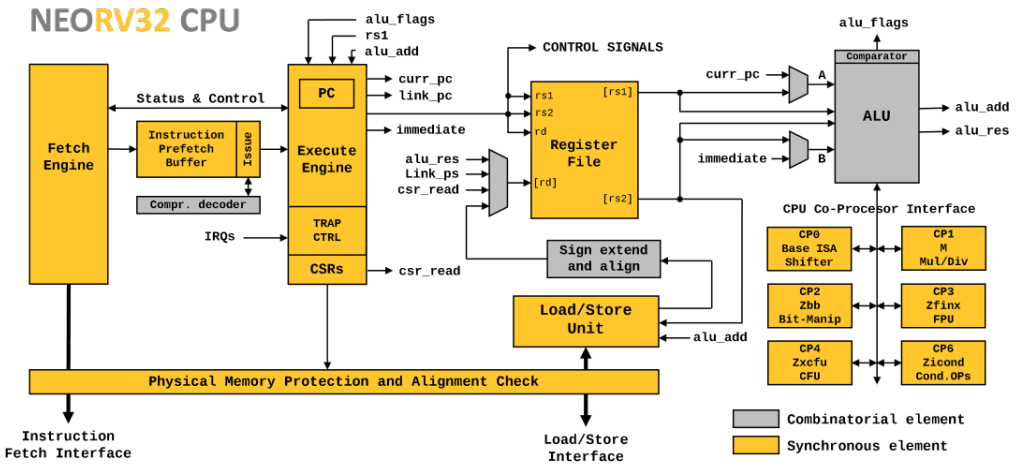


Figure 2.3: Detailed internal architecture of the NEORV32 CPU complex.

The internal microarchitecture of the NEORV32 CPU represents a sophisticated implementation of RISC-V principles, specifically optimized for FPGA resource efficiency and execution throughput.

Structurally, the processor is partitioned into four fundamental units, each designed to operate in a cohesive manner while maintaining a modular footprint:

- **CPU Control Unit**: Acts as the central orchestrator, overseeing instruction flow and generating control signals. It is logically divided into:
  - *Front-end*: Handles instruction fetching through a user-configurable *FIFO* prefetch buffer, enabling fetch and execution stages to run in parallel. It includes a simple branch prediction unit to minimize pipeline bubbles.
  - *Back-end*: Houses the architectural state, including the **Control and Status Registers (CSR)** and the trap controller, as well as the main hardware state machine.
- **BUS Unit**: Delegated to data management and memory interaction. It handles all load and store operations involving the Data Memory or peripheral address space. It is responsible for sub-word data adjustment (alignment and sign-extension) and incorporates the optional **Physical Memory Protection (PMP)** extension for hardware-level security.
- **Register File Unit**: Provides a synchronous storage bank of 32 general-purpose registers (reducible to 16 with the *RV32E* extension). It features two access ports for the simultaneous retrieval of operands and writing of results. Specific registers follow the RISC-V standard (e.g., $x0$ for zero). To save area, the Program Counter (PC) is integrated directly into the Control Unit, and the unit is designed to be mapped onto FPGA *Block RAM*.
- **ALU Unit**: The computational core for integer arithmetic and logical operations. It is highly modular, allowing the instantiation of the **M-extension** and a specialized floating-point unit (*F-extension*). Crucially, the ALU calculates jump and

branch target addresses, eliminating the need for additional adders elsewhere. Every extension operates with its own co-processor to manage internal signals and synchronization.

This streamlined integration is crucial for maintaining real-time determinism, a fundamental requirement for the high-frequency control loops used in UAV flight controllers.

2.4 Power Consumption in CMOS

To evaluate power-saving techniques in digital systems, it is necessary to identify the main sources of power consumption in CMOS circuits. This is particularly important in embedded and battery-powered platforms, where the available energy budget is limited and where the processor may represent a relevant contribution to the total system consumption.

The total power consumption of a CMOS circuit is commonly divided into two main components: static power and dynamic power. Static power is mainly caused by leakage currents that flow even when the circuit is not switching. It depends on the semiconductor technology, the supply voltage, the temperature, and the physical characteristics of the transistors. Dynamic power, instead, is associated with switching activity. Each time a node changes logic value, parasitic capacitances are charged or discharged.

For a synchronous digital circuit, a simplified expression of the dynamic power is:

$$P_{\text{dynamic}} = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f_{clk} \tag{2.1}$$

where:

- α is the switching activity factor;
- C_L is the effective switched capacitance;
- V_{dd} is the supply voltage;
- f_{clk} is the clock frequency.

This expression shows why clock activity is a central source of dynamic power in synchronous circuits. If a clock continues to propagate through registers and clock trees while no useful computation is being performed, dynamic power is still consumed. Consequently, reducing unnecessary switching activity through architectural techniques, such as clock gating, is one of the most direct ways to reduce dynamic power without changing the supply voltage or the nominal operating frequency [1].

In practical digital systems, total consumption results from the combination of static and dynamic power components [2]. Voltage scaling is also an effective technique, because of the quadratic dependence on V_{dd}. However, reducing the supply voltage also affects timing margins and can require a reduction in the maximum operating frequency to preserve correct operation. Therefore, any voltage reduction must be balanced against the corresponding performance impact [3].

In an FPGA-based experimental setup, voltage scaling is not always directly accessible or easy to control. For this reason, this work keeps the supply voltage and the system frequency fixed, and focuses instead on reducing switching activity through clock gating.

2.5 Power Saving Techniques

In order to reduce the power consumption of an electronic device, several power-saving techniques can be adopted at different levels of abstraction. Some techniques act on the supply voltage and clock frequency, others reduce leakage by disconnecting inactive blocks, while others aim to reduce the switching activity of clocked logic. The most common approaches considered in the context of digital processors and FPGA-based systems are Dynamic Voltage and Frequency Scaling, power gating, and clock gating.

1. **Dynamic Voltage and Frequency Scaling (DVFS):**

   Dynamic Voltage and Frequency Scaling (DVFS) is a power-management technique that adjusts a processor's supply voltage and operating frequency according to the current performance demand. When the computational workload is low, the operating frequency can be reduced, and the supply voltage can be lowered accordingly in order to decrease dynamic power consumption. When higher performance is required, the operating point can be increased again to provide the necessary throughput [4].

   The main advantage of DVFS is its direct impact on dynamic power. Since dynamic power depends approximately quadratically on the supply voltage and linearly on the clock frequency, reducing these two parameters can provide significant energy savings. DVFS is therefore particularly effective in systems where the workload varies over time and the processor does not always need to operate at its maximum performance point.

   However, DVFS also introduces important drawbacks. Reducing the supply voltage decreases the timing margin of the circuit, so the clock frequency must usually be reduced as well to preserve correct operation. This creates a trade-off between power reduction and performance. Moreover, DVFS requires additional hardware and software support, such as programmable voltage regulators, clock-generation circuitry, workload monitoring, and control policies able to decide when the operating point should be changed. These requirements increase the complexity of the system and make the technique less suitable for the simple FPGA-based experimental setup used in this thesis.

2. **Power Gating:**

   Power gating is a technique mainly used to reduce static power consumption. It works by disconnecting inactive circuit blocks from the supply rails, typically by means of sleep transistors. These transistors behave as high-impedance switches between the power supply and the logic block when the block is not needed [5].

   In a typical implementation, the sleep transistors can be placed either between the supply voltage and the circuit, acting as header switches, or between the circuit and ground, acting as footer switches. When the switches are turned off, the logic block is isolated from the power supply and leakage current is reduced. This can be very effective during long idle periods, especially in technologies where leakage power is a significant contribution to total power consumption.

   A major limitation of power gating is the possible loss of the internal state of the circuit. If a block is completely disconnected from the supply, the information stored in its registers or memory elements may be lost. To avoid this, additional retention elements or retention supply paths can be introduced, allowing the circuit to preserve its state while still consuming less power than in the fully active mode. These techniques increase the complexity of the design.

   Power gating also requires isolation logic, wake-up control, and careful management of the transition between powered and unpowered states. The wake-up time can be non-negligible, and the transition may involve current peaks due to recharging internal capacitances. For this reason, power gating is more suitable for relatively long inactivity periods, where the energy saved during the off time compensates for the overhead of entering and leaving the power-gated state.

3. **Clock Gating:**

   Clock gating is a power-saving technique used to reduce dynamic power consumption by selectively disabling the clock signal to functional blocks that are not currently required. In synchronous digital circuits, the clock network and the registers driven by it can consume power even when the data path is not performing useful computation. By preventing the clock from reaching inactive blocks, unnecessary switching activity can be reduced while keeping the circuit powered and preserving its state.

   Unlike power gating, clock gating does not disconnect the circuit from the supply rails. The logic remains powered, and the values stored in sequential elements are retained. This makes clock gating suitable for shorter idle intervals, where the system needs to resume operation quickly. For this reason, it is widely used in processors and embedded systems to reduce the energy wasted during idle or partially inactive phases.

   The effectiveness of clock gating depends on how much switching activity is avoided. If a large clock domain is disabled during a period in which it would otherwise keep toggling, the dynamic power reduction can be significant. However, the gating mechanism must be implemented carefully. A naive implementation based on ordinary logic gates can generate glitches or short unwanted clock pulses, which may lead to incorrect operation. For this reason, practical implementations rely either on dedicated clock-gating cells in ASIC flows or on dedicated clock-control primitives in FPGA devices.

   In FPGA-based designs, clock gating should not be implemented by routing a clock through generic LUT logic. FPGA vendors provide specific clocking resources for this purpose. In the Xilinx Artix-7 family used in this thesis, the relevant primitive is the BUFGCE, which provides a global clock buffer with an enable input and allows glitch-free control of a clock branch.

In this thesis, clock gating is selected as the main power-saving technique. DVFS is not used because the experimental setup keeps the supply voltage and system frequency fixed, and because voltage-frequency control would require additional board-level and software support. Power gating is also not adopted, since it is more invasive, usually targets leakage reduction, and is not directly available for arbitrary logic regions in a standard FPGA fabric. Clock gating, instead, can be implemented using the dedicated BUFGCE primitive available in Xilinx Artix-7 devices. This makes it suitable for the objective of this work: reducing dynamic switching activity in the NEORV32 CPU clock domain during `WFI`-based sleep intervals while keeping the wake-up infrastructure active.

2.6 Clock Gating Basics

Clock gating is a technique used to reduce dynamic power consumption by preventing the clock signal from reaching circuit blocks that are not active. In synchronous systems, a large portion of switching activity is caused by the clock network and by the sequential elements driven by it. Even when a functional block is not processing useful data, its registers may still receive clock edges and toggle internally. Disabling the clock during inactive periods therefore reduces unnecessary transitions and lowers dynamic power consumption [3].

The simplest conceptual form of clock gating is the latch-free approach, in which the clock signal is combined with an enable signal through a logic gate, typically an AND gate. When the enable signal is high, the clock propagates to the downstream circuit. When the enable signal is low, the gated clock remains at zero and the downstream sequential logic stops receiving clock edges. This basic structure is illustrated in Fig. 2.4.

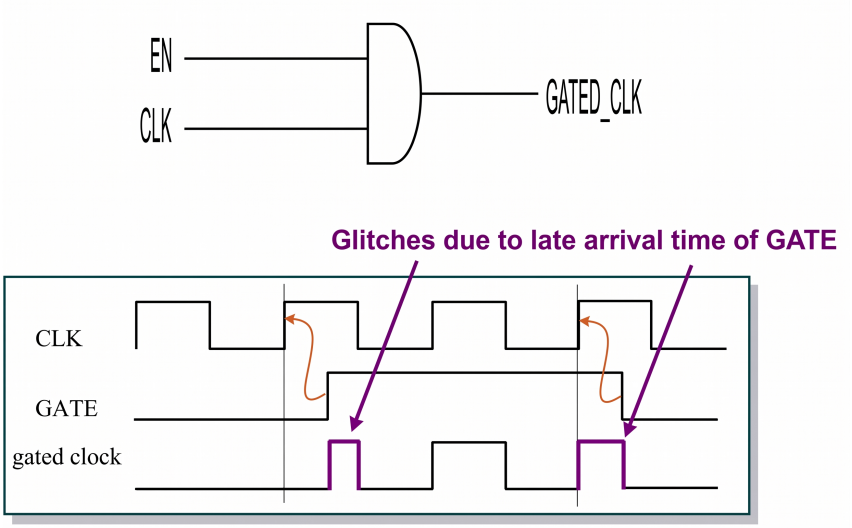


Figure 2.4: Latch-free clock gating concept.[6]

Although this structure is simple, it has an important limitation. The enable signal must remain stable during the sensitive portion of the clock period. If the enable signal changes while the clock is high, or close to an active clock edge, the output of the gate may contain a partial pulse or a glitch. Such unwanted clock pulses can trigger flip-flops incorrectly, violate timing constraints, and increase rather than reduce switching activity.

For this reason, simple combinational clock gating is generally not suitable for robust digital designs, especially in FPGA implementations. In an FPGA, routing the clock through ordinary LUT fabric is discouraged because it bypasses the dedicated clocking resources of the device and may introduce skew, glitches, and timing closure issues.

In this thesis, the latch-free structure is considered only as a conceptual introduction to clock gating. The actual implementation does not use LUT-based gating. Instead, the CPU clock is controlled through the Xilinx BUFGCE primitive, which is specifically designed to enable and disable clock propagation in a glitch-free way on Artix-7 FPGAs.

2.7 Latch-Based Clock Gating Technique

Latch-based clock gating was introduced to overcome the main weakness of simple latch-free clock gating. In the latch-free approach, the enable signal is applied directly to the clock through a logic gate. If this enable signal changes at an unsafe time, the gated clock can contain glitches or short unwanted pulses. Latch-based clock gating avoids this issue by inserting a latch that stores the enable signal during the safe portion of the clock period.

The latch holds the enable value stable while the clock is in the phase where a transition could generate a glitch. As a consequence, the gated clock is either fully propagated or fully blocked, instead of producing partial pulses. This makes latch-based clock gating

more robust than a simple AND-based solution and is one of the reasons why dedicated clock-gating cells in ASIC design commonly include latch-based structures [2].

Figure 2.5 shows the principle of latch-based clock gating. The enable signal is first captured by the latch and then used to control the gated clock. By preventing the enable signal from changing during the critical portion of the clock period, the circuit reduces the risk of unwanted pulses.

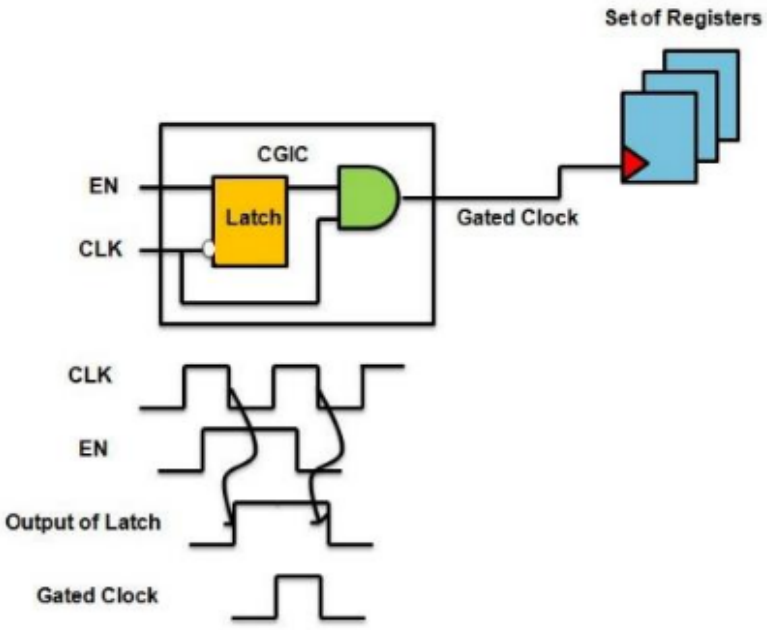


Figure 2.5: Latch-based clock gating concept used to avoid glitches.[2]

Although latch-based gating is a common concept in ASIC design, this thesis does not implement a custom latch-based clock-gating cell in the FPGA fabric. In FPGA designs, clock signals should be routed through the dedicated clocking resources provided by the device vendor. For this reason, the latch-based technique is presented here only as a conceptual step between simple combinational gating and the dedicated FPGA primitive used in this work. The actual implementation relies on the Xilinx BUFGCE primitive, which provides glitch-free clock enable control for the selected CPU clock domain.

2.8 Hardware Primitives for Clock Gating

In FPGA implementations, clock gating should be performed using dedicated clocking resources rather than ordinary logic fabric. Routing a clock through LUTs or generic combinational gates can introduce excessive skew, glitches, and timing closure problems. For this reason, FPGA vendors provide specific clock-control primitives that are designed to safely enable, disable, or route clock signals.

In Xilinx 7 Series FPGAs, such as the Artix-7 devices used in this thesis, several clocking resources can be used depending on the required clocking scope. Examples include `BUFGCE`, `BUFHCE`, and `BUFR` [7]. The choice of the buffer depends on whether the clock must drive a global domain, a regional part of the device, or a more localized portion of the design.

2.8.1 The BUFGCE Primitive

The primitive selected for this work is the `BUFGCE`, a global clock buffer with an active-high clock enable input [8]. This primitive is derived from the more general `BUFGCTRL` structure and is intended to provide glitch-free control of a global clock branch. The BUFGCE has three main ports, as shown in Fig. 2.6:

1. `I`: input clock signal;
2. `CE`: active-high clock enable signal;
3. `O`: output clock signal.

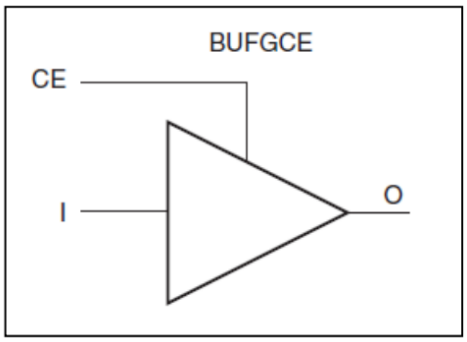


Figure 2.6: BUFGCE primitive symbol. [8]

When `CE` is asserted, the output clock `O` follows the input clock `I`. When `CE` is de-asserted, the output clock is held inactive, preventing clock edges from reaching the downstream clock domain [8]. The logical behavior and port interface of the primitive are summarized in Table 2.1.

Table 2.1: Logical operation and port interface of the BUFGCE primitive.

Port	Direction	Width	Function
`CE`	Input	1	Active-high clock enable signal.
`I`	Input	1	Primary clock input.
`O`	Output	1	Gated clock output.
Logic state: if `CE`=0 then `O`=0; if `CE`=1 then `O`=`I`.			

In the proposed NEORV32 implementation, the BUFGCE output is used to generate the dedicated CPU clock, named `clk_cpu`. The enable input is driven by a condition derived from the processor sleep state and from the wake-up sources required to resume execution. This makes it possible to suppress the CPU clock during `WFI`-based sleep intervals while keeping the timer, UART, and measurement logic active on the global clock.

2.8.2 Buffer Selection and Justification

The Artix-7 clocking architecture provides different buffering resources, each suited to a specific use case [7]. The main alternatives considered in this work are:

- `BUFR`: a regional clock buffer mainly intended for regional clocking and I/O-related clock management. It is suitable when the clock domain is limited to a small area of the device, but it is not ideal for a processor subsystem that may span multiple regions.
- `BUFHCE`: a horizontal clock buffer with clock enable. It can be used to gate clocks within a clock region and may provide fine control over localized logic. However, its scope is more limited than a global clock buffer.
- `BUFGCE`: a global clock buffer with clock enable. It can drive a large clock domain across the FPGA fabric and is therefore suitable for coarse-grain clock gating of a processor-related clock domain.

For this thesis, `BUFGCE` was selected because the gated domain includes the NEORV32 CPU and part of the surrounding processor infrastructure. These blocks are not guaranteed to be confined to a single small region of the FPGA after placement and routing. A global clock buffer therefore provides a safer and more flexible solution for this type of coarse-grain gating.

The goal of the proposed architecture is not to gate an isolated functional unit, but to suppress the CPU-side clock domain during sleep intervals triggered by the RISC-V `WFI` instruction. The `BUFGCE` primitive provides a glitch-free and FPGA-supported mechanism to implement this behavior, avoiding the risks associated with LUT-based clock gating and preserving compatibility with the FPGA timing flow.

Chapter 3

State of the Art

In this chapter, a comprehensive overview of the existing literature regarding energy-efficiency optimizations in embedded processors is performed, with particular attention given to Register-Transfer Level (RTL) techniques applied to the RISC-V architecture and their implementation on Field-Programmable Gate Arrays (FPGAs). The growing demand for battery-powered embedded systems, particularly in the Internet of Things (IoT) and Unmanned Aerial Vehicles (UAVs), has made power efficiency a primary design concern. As highlighted by Ogundairo and Anggreani [9], decisions made at the RTL stage have a foundational impact on the final chip's Power, Performance, and Area (PPA) metrics.

Among the various strategies for energy reduction, clock gating remains one of the most effective methods for decreasing dynamic power dissipation in synchronous circuits. A foundational analysis of clock gating behavior in sequential circuits was proposed by Wu et al. [10], who modeled clock transitions to selectively drive flip-flops based on quaternary variables. This line of research has been further expanded in recent overviews, such as the one provided by Chindhu and Shanmugasundaram [11], who categorized various techniques into latch-based and data-driven gating, identifying them as essential components for reducing the energy wasted by the system clock signal.

Specifically within the RISC-V ecosystem, Santhosh and Deshpande [12] have recently demonstrated a low-power implementation using fine-grained clock and power gating. Their work explores the selective disabling of functional units during idle cycles, which is particularly beneficial for pipelined cores where not all modules are active simultaneously. Focusing on the computational core, Vo [2] proposed a hybrid data-driven clock and data gating technique specifically for the RISC-V ALU, achieving up to a 46.67% reduction in power consumption compared to original designs. Similarly, Zhao and Wang [13] explored improvements in gating circuits to prevent glitches and "burrs" through data-holding modules, utilizing the ISCAS89 benchmark to indicate that dynamic consumption can be reduced by more than 20%.

Complementing these RTL-level strategies, the underlying semiconductor technology plays a critical role in overall power optimization. Ahmed et al. [5] highlight that as process technologies scale, structures such as Fully Depleted Silicon-on-Insulator (FD-SOI) and Fin Field Effect Transistors (FinFET) are replacing bulk MOSFETs to address leakage issues. Their research indicates that while FinFET-based designs can consume significantly less energy during active modes, FDSOI remains superior for high-speed operations due to lower propagation delays. Furthermore, they propose advanced power gating techniques using Data Retention Transistors to maintain logical states during "Hold

Modes," creating virtual supply rails that significantly reduce static power without losing data.

Nevertheless, despite these advancements, most studies are frequently conducted using ASIC-oriented simulation platforms or generic benchmarks, which may not accurately reflect the behavior of commercial FPGAs under realistic workloads. The implementation on commercial FPGA platforms introduces unique challenges related to hardware primitives and timing closure. Tan et al. [1] conducted a study specifically on the Xilinx Artix-7 FPGA—the same family used in this thesis—implementing clock gating on a 32-bit RISC processor and achieving a 24% reduction in clock tree power. This research is supported by the work of Bezati et al. [14], [15], who explored coarse-grain clock gating for streaming applications, demonstrating that power savings can be integrated into high-level dataflow design flows without loss in data throughput.

A significant research gap remains, however, as these FPGA-centric studies often stop at software-based power estimations provided by tools like Xilinx Vivado [16], without bridging the gap to high-precision physical hardware-in-the-loop measurements. The maturation of the RISC-V ecosystem has recently enabled more sophisticated evaluation and debugging frameworks to address this. The NEORV32 core, developed by Nolting et al. [17], provides a highly optimized VHDL baseline that has become a reference for modular SoC design. Parallelly, the work of Kojima et al. [18] has introduced open-source frameworks for efficient side-channel analysis, enabling modern FPGA design patterns on boards such as the CW305. This framework, along with the methodologies proposed by Dewar et al. [19] using the ChipWhisperer platform, represents the current frontier for validating the physical power characteristics of hardware implementations.

While the aforementioned literature provides extensive insights into individual gating techniques, there is a lack of a unified study that combines fine-grained RTL optimizations on the NEORV32 core with high-precision physical power trace validation specifically for UAV control loops. Generic benchmarks often do not account for the specific mission profiles of UAV controllers, such as frequent idle-polling phases, nor do they characterize how these modifications impact the real-time determinism required for flight stability. This thesis aims to address this gap by leveraging the modularity of RISC-V and the precision of side-channel analysis tools to provide a physical "ground truth" for energy efficiency in autonomous aerial systems.

Chapter 4

Proposed Approach and Methodology

This chapter describes the proposed clock-gating strategy and the methodology adopted to validate it on an FPGA-based NEORV32 system. The objective is to connect a standard RISC-V software-visible sleep mechanism, namely the `WFI` instruction, to a hardware clock-gating structure implemented with FPGA-safe clocking resources.

The approach is based on a hardware/software co-design flow. On the hardware side, the NEORV32 top-level is modified to introduce a dedicated CPU clock domain controlled by a Xilinx BUFGCE primitive. On the software side, the CoreMark benchmark is instrumented to alternate active computation windows with timer-driven sleep intervals. This makes it possible to compare continuous execution, WFI-only behavior, a permanently enabled clock-path control case, and the fully clock-gated implementation.

The methodology is designed to isolate the contribution of actual clock suppression from other effects, such as reduced software activity during `WFI` or routing changes caused by inserting the BUFGCE primitive. For this reason, the validation includes control configurations and physical measurements on FPGA hardware, rather than relying only on tool-based power estimation.

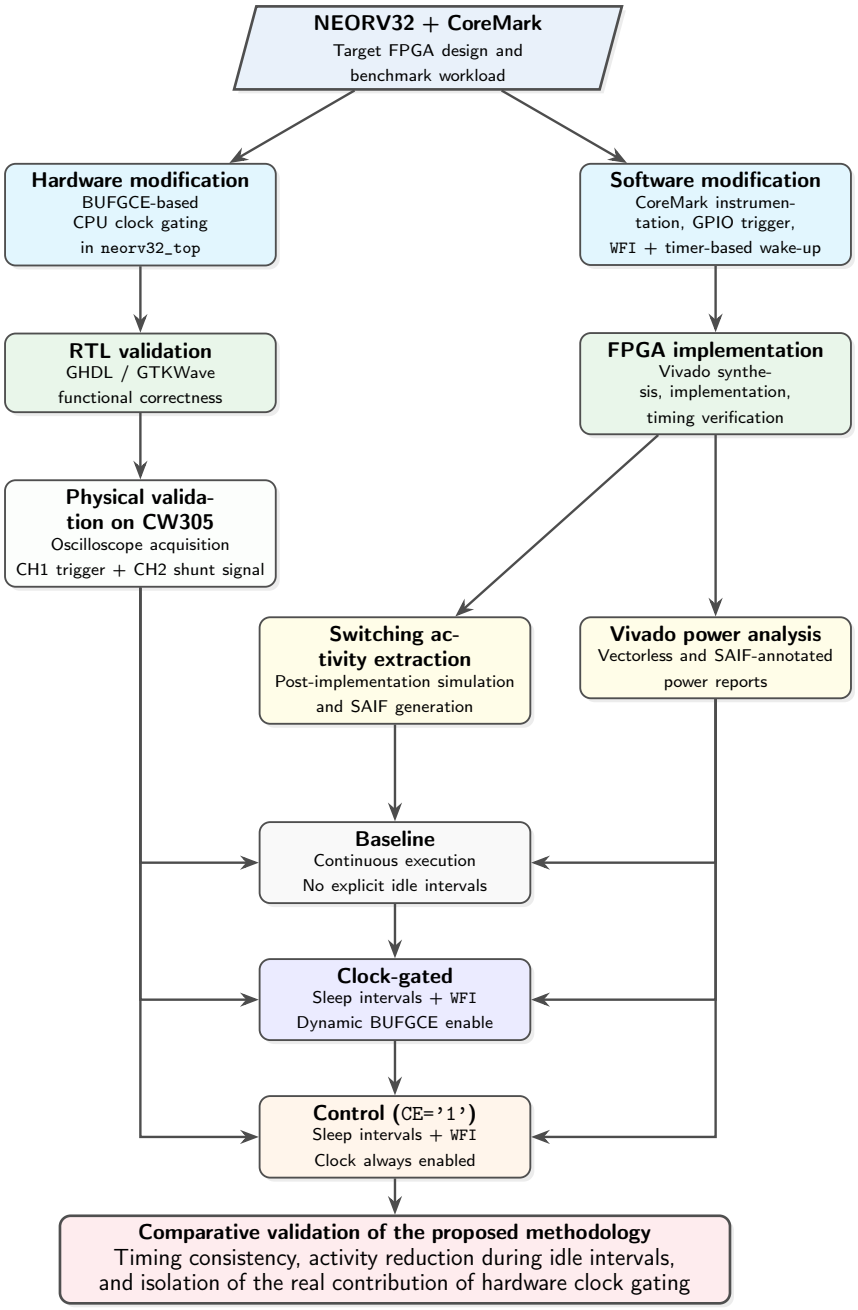


Figure 4.1: Technical flow of the proposed methodology. The approach combines hardware and software modifications, RTL and implementation-level validation, SAIF-based power estimation, and physical oscilloscope measurements on the CW305 platform. The final comparison among baseline, clock-gated, and `CE='1'` control configurations is used to isolate the actual contribution of hardware clock suppression.

4.1 The Research Gap: Clock Gating in UAV-Relevant Embedded Systems

While clock gating is a well-documented technique for general-purpose processors [10], its application to FPGA-based RISC-V embedded controllers in UAV-relevant scenarios still requires careful investigation. Current literature often focuses on ASIC-oriented designs [12] or on generic computational benchmarks, such as the ISCAS89 circuits used by Zhao

and Wang [13] or the portable computer workloads discussed by Sulaiman [3]. These works are useful to demonstrate the potential of clock gating, but they do not fully address the behavior of a complete FPGA-based soft-core processor executing software with explicit sleep and wake-up phases.

Unlike purely synthetic workloads, UAV control systems and similar embedded applications are often interrupt-driven. They may alternate between short computation windows and waiting periods for timers, sensor updates, communication events, or control deadlines. As highlighted in the PPA analysis for RISC-V SoCs [9], RTL-level decisions are critical because they directly influence power, performance, and area. In this context, the key issue is not only whether the processor stops executing instructions, but also whether the clock continues to propagate through the sequential logic during idle periods.

Furthermore, UAV-like workloads may include frequent idle-polling or waiting phases while the processor is waiting for high-frequency sensor data, such as IMU or GPS updates. This differs from continuous, instruction-heavy execution patterns commonly used in generic benchmarks such as CoreMark or Dhrystone. While studies on commercial FPGAs [1] have shown that clock gating can reduce dynamic power, tool-based estimates alone are not sufficient to isolate the real contribution of clock suppression. A measured reduction may be caused by software sleep, modified routing, or the actual removal of clock transitions. For FPGA implementations, this distinction is especially important because clock gating must be implemented through dedicated clocking resources such as the `BUFGCE` primitive [8], rather than through simple logic-based gating.

This thesis aims to address this gap by focusing on three critical aspects:

- **UAV-relevant workload behavior:** Most studies evaluate power savings using generic mathematical loops or benchmark circuits targeting specific functional units, such as the ALU [2], or standardized circuit benchmarks such as the ISCAS89 suite [13]. Other research focuses on general-purpose RISC-V implementations on FPGA [1], [12] without explicitly separating active computation from software-defined waiting intervals. In this work, CoreMark is not used as a complete UAV workload, but as a controlled and repeatable benchmark that can be divided into active computation windows and timer-driven sleep intervals. This makes it possible to reproduce an active/idle structure that is relevant to interrupt-driven embedded controllers.

- **Physical validation versus software estimation:** Reliance on software-based power reports, such as those produced by Xilinx Vivado [16], can lead to incomplete conclusions because these tools depend on probabilistic models or on switching-activity files that may not capture all board-level and measurement-chain effects. As highlighted by Kojima et al. [18], physical power measurements are important for validating the behavior of real FPGA implementations. This work therefore combines tool-based analysis with physical measurements, using the ChipWhisperer CW305 platform [19] and the Arty A7-100T XADC-based acquisition flow to obtain complementary evidence.

- **Controlled isolation of the clock-gating effect:** In FPGA-based systems, a reduction in measured activity during idle intervals may come from several sources: the execution of `WFI`, a change in software activity, the insertion of the BUFGCE clock path, placement and routing changes, or the actual suppression of clock transitions. For this reason, the proposed validation includes a `CE = 1` control configuration, where the BUFGCE remains instantiated but the clock enable is forced

permanently high. This makes it possible to distinguish the effect of software-level sleep from the additional reduction obtained through real hardware clock suppression. The use of the `BUFGCE` primitive [8] also ensures that the gating mechanism is implemented through an FPGA-supported clocking resource, avoiding the risks associated with LUT-based clock gating.

4.2 Standard-Compliant Strategy: The WFI Trigger

A key design choice of this work is to use the standard RISC-V `WFI` (Wait-For-Interrupt) instruction as the software-visible trigger for the low-power state. The objective is to avoid introducing a custom, non-portable software interface for clock gating. Instead, the processor enters the low-activity interval through a standard ISA mechanism that is already intended to suspend execution until an interrupt or wake-up event occurs.

In the proposed implementation, `WFI` is used inside the instrumented CoreMark flow. After a benchmark chunk has been executed, the software programs the machine timer compare register and then executes `neorv32_cpu_sleep()`, which relies on `WFI`. The machine timer interrupt is then used as the wake-up source. This creates a repeatable active/sleep pattern that can be observed both in simulation and in physical measurements.

Using `WFI` as the trigger provides several advantages:

- **ISA compliance:** The sleep request is generated through a standard RISC-V instruction rather than through a custom memory-mapped register or a non-standard instruction. This keeps the mechanism compatible with the RISC-V programming model and makes the approach easier to reproduce on other RISC-V implementations.
- **Clear hardware/software boundary:** The software is responsible for deciding when the processor can enter a sleep interval, while the hardware is responsible for translating the processor sleep state into safe clock control. In this work, the hardware side uses the NEORV32 `cpu_sleep` signal to drive the BUFGCE-based clock enable logic.
- **Repeatable experimental timing:** By combining `WFI` with the machine timer interrupt, the duration of the sleep interval can be controlled in software. This makes it possible to generate repeatable active/sleep sequences during CoreMark execution and to compare the baseline, WFI-only, CE = 1, and clock-gated configurations under similar timing conditions.
- **Safe wake-up path:** The clock cannot simply be disabled without preserving the logic required to wake the processor. For this reason, the timer and interrupt infrastructure must remain active while the CPU clock domain is gated. In the implemented design, the CLINT timer, UART interface, and measurement logic are kept on the global clock, while the CPU-side domain is driven by the gated clock.

This strategy makes the clock-gating mechanism dependent on a standard architectural sleep event while still allowing the FPGA hardware to suppress clock activity during the corresponding idle interval. In this way, the implementation avoids a purely software-level sleep solution and enables the experimental separation between `WFI`-only behavior and actual hardware clock suppression.

4.3 Hardware Implementation for Real-Time Performance

In embedded controllers, low-power states are useful only if the processor can resume execution in a predictable way. This is especially relevant for interrupt-driven systems, where the processor may wait for timers, sensor events, or communication interfaces and must resume operation when the next event becomes available.

For this reason, the proposed clock-gating mechanism is designed so that the CPU clock can be disabled only while the processor is in a valid sleep state, and re-enabled when a wake-up source becomes pending. The objective is not to introduce a deep power-down state with long recovery time, but to reduce dynamic switching activity during short software-defined idle intervals while preserving a deterministic interrupt-driven wake-up path.

In the implemented NEORV32 system, the sleep state is reached through the standard RISC-V `WFI` instruction. Before entering sleep, the software programs the machine timer compare register. When the timer expires, the machine timer interrupt becomes pending and the CPU clock is re-enabled. This allows the processor to resume execution after the programmed sleep interval.

A critical design choice is that not all modules are placed in the gated clock domain. If the timer or interrupt logic were gated together with the CPU, the system could enter a deadlock: the processor would stop its clock, but the wake-up source would no longer be able to generate the event required to restart it. For this reason, the CLINT timer, UART communication path, and measurement logic remain connected to the global clock. Only the CPU-side domain is driven by the gated clock generated through the BUFGCE primitive.

This architecture preserves the wake-up infrastructure while reducing the switching activity of the CPU-related logic during sleep intervals. The approach is therefore suitable for evaluating short active/sleep sequences such as those generated by the instrumented CoreMark benchmark used in this work.

4.3.1 Synchronous Gating via BUFGCE

The proposed design integrates the AMD/Xilinx `BUFGCE` global clock buffer with clock enable [7]. This primitive is used to generate a dedicated clock signal for the CPU-side domain, named `clk_cpu`. The input of the `BUFGCE` is connected to the global system clock, while its output drives the logic that is intended to be stopped during sleep intervals.

The `BUFGCE` does not disable the complete FPGA clock distribution. It only stops the clock branch connected to its output. For this reason, the proposed design keeps the wake-up and communication infrastructure, such as the CLINT timer, UART, and measurement logic, connected to the global clock. This avoids sleep deadlock and allows the processor to resume execution when a timer or interrupt event becomes pending.

- **Deterministic wake-up behavior:** The clock is restored through the FPGA clocking network when the enable signal is asserted again. The wake-up path therefore remains interrupt-driven and does not rely on software polling or on an unsafe clock restart mechanism.
- **Glitch-free transitions:** Unlike LUT-based gating, where the clock would be masked using ordinary FPGA fabric, the `BUFGCE` is a dedicated hard primitive.

This avoids runt pulses and malformed clock edges that could otherwise occur if a combinational enable signal changed at an unsafe point of the clock cycle.

This makes the `BUFGCE` suitable for the proposed coarse-grain gating strategy: the CPU-side clock domain can be suppressed during `WFI`-based idle intervals, while the rest of the system remains able to provide wake-up, communication, and measurement support.

4.3.2 Portability Across FPGA Technologies

Although the implementation presented in this thesis targets Xilinx Artix-7 devices, the underlying idea is not limited to this FPGA family. The proposed strategy is based on two separate concepts: using the standard RISC-V `WFI` instruction to identify software idle intervals, and mapping this idle condition to a vendor-supported clock-control primitive.

The first part is architecture-independent. The `WFI` instruction belongs to the RISC-V ISA and can therefore be used as a standard software-visible indication that the processor is entering a low-activity state. The second part, instead, is technology-dependent: the actual clock-gating element must be selected according to the clocking resources available in the target FPGA family.

In Xilinx 7 Series devices, this role is performed by the `BUFGCE` primitive, which provides glitch-free control of a global clock branch. Other FPGA vendors provide conceptually similar resources. For example, Intel/Altera devices provide clock-control blocks such as `ALTCLKCTRL`, while some Lattice FPGA families provide dynamic clock-control primitives such as `DCC` or related clock-gating resources. These blocks are not identical to `BUFGCE`, and their timing behavior, routing scope, and enable semantics must be verified separately for each target technology.

For this reason, the results obtained in this thesis should be interpreted as a validation of the method on Xilinx Artix-7 platforms, not as a direct experimental proof for all FPGA families. Nevertheless, the methodology is portable at architectural level: a standard RISC-V sleep event can be connected to a safe vendor-specific clock-control primitive, as long as the wake-up infrastructure remains active and the gated domain is defined carefully.

4.4 Multi-Layered Validation Flow

The validation flow adopted in this work combines simulation, implementation-level power analysis, and physical measurements. This multi-layered approach is necessary because no single validation method is sufficient to fully characterize the behavior of the proposed clock-gating strategy.

At RTL level, simulation is useful to verify the logical behavior of the modified design. It allows checking that the clock-gating control reacts to the processor sleep state and that wake-up events correctly restore execution. However, RTL simulation alone does not provide reliable information about physical power reduction, since it does not include the actual FPGA clock network, routing, board-level effects, or measurement-chain behavior.

For this reason, the design is also evaluated through the FPGA implementation flow. Vivado synthesis, implementation, timing analysis, and power estimation are used to verify that the modified NEORV32 system can still be mapped on the target FPGA and that the insertion of the gated clock path does not break timing closure. In addition,

SAIF-based power analysis is used to improve the switching-activity information provided to the power estimator with respect to purely vectorless estimates.

Finally, the validation is extended to physical measurements on FPGA hardware. The CW305 platform is used to observe the shunt-related activity signal with an oscilloscope, providing a direct time-domain view of active and idle intervals. The Arty A7-100T platform is used as a second measurement setup based on the on-board current-sense path and the FPGA XADC. In this case, the acquisition is integrated into the design and the samples are exported through UART for MATLAB post-processing.

The combination of these levels makes the validation more robust. Simulation and Vivado analysis verify that the implementation is functionally and structurally correct. The CW305 measurements provide physical evidence of activity modulation on the supply path. The Arty A7-100T measurements provide a more automated and phase-aware comparison between active, sleep, and control configurations.

4.4.1 Logical Correctness via RTL Simulation

The first validation step consists of checking the logical correctness of the modified NEORV32 design. RTL simulation is used to verify that the processor can enter and leave sleep states without reaching illegal conditions and that the clock-gating control behaves consistently with the intended active and idle phases.

In particular, the relevant signals include the processor sleep indication, the timer interrupt used for wake-up, and the gated clock enable. The objective of this step is not to estimate the final power consumption, but to ensure that the hardware modification is functionally meaningful before moving to FPGA implementation and physical measurements.

4.4.2 Empirical Validation via Physical Measurements

The final validation step relies on physical measurements performed on FPGA boards. The CW305 platform is used to obtain oscilloscope-based measurements of the shunt-related activity signal. By acquiring the trigger signal and the power-related waveform simultaneously, it is possible to visually correlate benchmark execution with changes in the measured activity.

A second physical validation is performed on the Arty A7-100T platform using the on-board current-sense path and the FPGA XADC. This setup does not provide the same analog bandwidth as an oscilloscope-based measurement, but it enables a more automated and repeatable acquisition flow. The samples are collected by hardware, associated with their execution phase, and processed offline in MATLAB.

Using both platforms provides complementary evidence. The CW305 setup offers a direct view of the activity waveform, while the Arty setup enables phase-wise statistics over multiple CoreMark iteration counts. Together, they support the interpretation that the measured reduction is caused by the combination of `WFI`-based sleep intervals and actual CPU clock suppression.

4.5 Implementation and Simulation-Based Power Analysis Workflow

Before moving to physical measurements, the NEORV32 system had to be made stable in the FPGA implementation flow. This phase was necessary because the clock-gating strategy could not be evaluated meaningfully until the CoreMark application was correctly implemented, simulated, and mapped on the target FPGA.

The workflow served two main purposes. First, it provided a working NEORV32/CoreMark baseline implementation that could be used as a reference for the modified designs. Second, it established a repeatable Vivado-based analysis flow including synthesis, implementation, post-implementation timing simulation, switching-activity extraction, and power estimation.

This workflow was not limited to the baseline case. Once the CoreMark image and the internal memory configuration were corrected, the same implementation flow could be used for the baseline configuration, for the clock-gated implementation, and for the control cases used to evaluate the effect of the BUFGCE clock path.

The Vivado power reports obtained in this phase were used as simulation-based support for the design process. Vectorless reports provided a preliminary estimate based on default activity assumptions, while SAIF-annotated reports used workload-dependent switching activity extracted from simulation. These reports were not treated as the final experimental evidence of the thesis; the final validation was performed through physical measurements on the CW305 and Arty A7-100T platforms.

4.5.1 CoreMark Image Issue and Memory Sizing

The memory-sizing problem emerged during the transition from the initial software/RTL tests to the Vivado FPGA implementation flow. CoreMark is not a minimal test program: it includes data structures and benchmark routines that generate a non-negligible program image. During the first tests, a simulation-oriented memory configuration was used, with

$$\texttt{IMEM\_SIZE} = 512 \text{ KB}, \qquad \texttt{DMEM\_SIZE} = 64 \text{ KB}.$$

This configuration was convenient during the early RTL and software validation phase, because it avoided program-image overflow and allowed the CoreMark image to be loaded without immediately hitting memory-size limitations. However, when moving to the Vivado FPGA implementation flow, this setup became unnecessarily heavy for the target design.

The main issue was related to the generated instruction-memory image file, `neorv32_imem_image.vhd`. In the initial flow, this file contained not only the actual useful program contents, but also a large unused region filled with zero words. As a consequence, the generated VHDL memory image was much larger than the portion of memory actually required by the CoreMark program.

This caused practical problems when trying to run the design through the Vivado-based implementation and simulation flow. Allocating a much larger internal instruction memory than required increased the memory footprint and made the design unnecessarily heavy for the target implementation. As a result, before evaluating the clock-gating mechanism, the program-image generation and the internal memory sizes had to be corrected.

The solution was to modify the image-generation tool, `imem_image_gen.c`, so that the emitted instruction-memory image ends at the last non-zero word. In this way, the generated `neorv32_imem_image.vhd` file contains the actual CoreMark program contents without carrying a large unused zero-filled tail.

After this correction, the useful program image was reduced to approximately

$$33 \text{ KB}.$$

This made it possible to move to the final internal memory configuration used for the Arty A7-100T implementation:

$$\texttt{IMEM\_SIZE} = 64 \text{ KB}, \qquad \texttt{DMEM\_SIZE} = 32 \text{ KB}.$$

The final memory configuration was therefore not selected as an independent power-optimization strategy. It was a practical correction required to obtain a working and manageable Vivado implementation and simulation flow. The reduction in unused memory allocation is a beneficial consequence of this correction, because it avoids allocating FPGA memory resources for a zero-filled region that is not required by the actual CoreMark workload.

Table 4.1: IMEM/DMEM configurations considered during the CoreMark implementation flow.

Configuration	**IMEM**	**DMEM**	**Use**
Initial simulation-oriented setup	512 KB	64 KB	Early RTL/software validation
Final Arty A7-100T implementation	64 KB	32 KB	Vivado implementation flow

The table summarizes the two memory configurations used during the development flow. The initial configuration was useful for early validation, while the final configuration was adopted after correcting the generated IMEM image. The final setup supports the complete CoreMark workload while keeping the internal memory footprint manageable.

4.5.2 Vivado Implementation and Timing Flow

After correcting the CoreMark image generation and selecting a manageable internal memory configuration, the NEORV32/CoreMark design was processed through the Vivado FPGA implementation flow. This step was required before any simulation-based power analysis or physical measurement could be considered meaningful, because the design first had to be synthesizable, routable, and timing-clean on the target FPGA.

The Vivado implementation flow consists of synthesis, placement, routing, and timing verification. Synthesis translates the RTL description into a gate-level netlist, while implementation maps the processor, internal memories, GPIO, UART, timer logic, XADC monitoring logic, and clocking resources onto the Artix-7 FPGA fabric.

Timing verification is a critical step in this work because the proposed architecture modifies the clocking structure of the processor. The insertion of a BUFGCE-based CPU clock path must not break timing closure or introduce unconstrained internal paths. For this reason, after implementation the timing summary was checked before proceeding to switching-activity extraction and power estimation.

The implementation flow was first applied to the baseline configuration, which provided the initial working NEORV32/CoreMark design. The same flow was then used for the modified configurations, including the WFI-only, CE = 1, and clock-gated designs. This ensures that all configurations are analyzed through a consistent implementation flow.

Figure 4.2 shows the implemented design view for the baseline NEORV32/CoreMark configuration on the Arty A7-100T platform. Figure 4.3 reports the corresponding timing summary after implementation. The timing report confirms that the design satisfies the selected 100 MHz clock constraint before moving to power estimation.

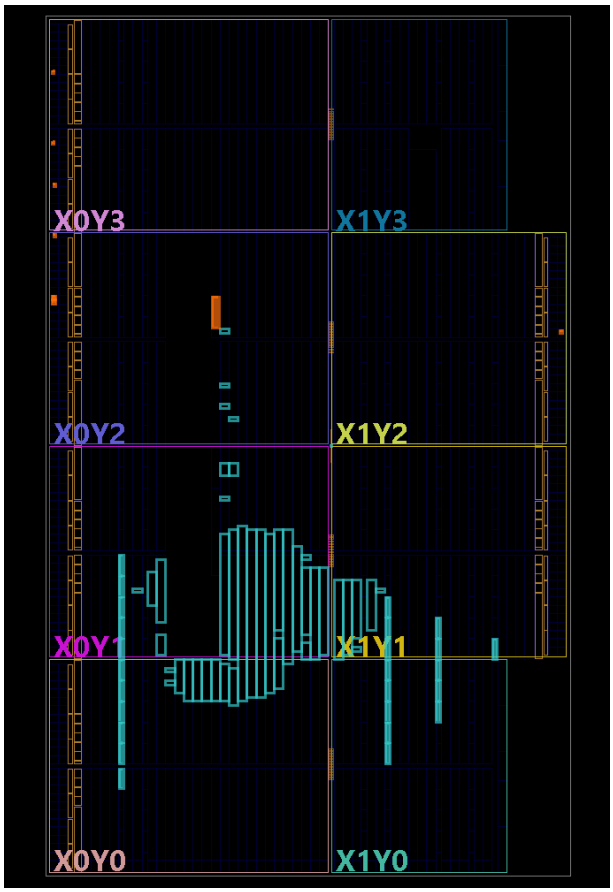


Figure 4.2: Implemented design view for the baseline NEORV32/CoreMark configuration on the Arty A7-100T platform.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1,110 ns	Worst Hold Slack (WHS):	0,037 ns	Worst Pulse Width Slack (WPWS):	3,750 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	5965	Total Number of Endpoints:	5965	Total Number of Endpoints:	2195

All user specified timing constraints are met.

Figure 4.3: Timing summary after implementation for the baseline NEORV32/CoreMark configuration on the Arty A7-100T platform.

4.5.3 Vectorless and SAIF-Annotated Power Analysis

After implementation and timing verification, Vivado was used to perform a simulation-based power analysis. This step provides an intermediate validation level between pure RTL simulation and physical board measurements. Its purpose is not to replace the measurements on CW305 and Arty A7-100T, but to verify whether the implemented design shows a consistent power trend at tool level.

Vivado can estimate switching activity in two main ways:

- *Vectorless/default activity*: Vivado propagates switching activity from constraints and internal default assumptions. This method can be used without a simulation activity file and is useful for obtaining a preliminary power estimate. However, it is not workload-specific and therefore cannot accurately represent the activity generated by CoreMark, UART traffic, memory accesses, or WFI-based sleep intervals.
- *SAIF-annotated activity*: Vivado uses a Switching Activity Interchange Format file generated from simulation. In this case, the power report is annotated with workload-dependent switching activity, making the estimate more meaningful for comparing baseline execution, WFI-only behavior, CE = 1 control, and dynamic clock gating.

As a first reference, a vectorless power report was generated for the routed baseline configuration. Since no simulation activity file was provided, the switching activity is not specific to the CoreMark execution. The report is therefore used only as a preliminary reference, as also indicated by the low confidence level.

Table 4.2: Vivado vectorless power estimate for the routed baseline configuration.

(a) Power summary.

Quantity	**Value**
Device	xc7a100tcsg324-1
Design state	Routed
Total on-chip	0.131 W
Dynamic	0.033 W
Device static	0.098 W
Junction temp.	25.6 °C
Max ambient	84.4 °C
Effective θ_{JA}	4.6 °C/W
SAIF file	–
Confidence	Low

(b) Dynamic breakdown.

Component	**Power**
Clocks	0.011 W
Slice logic	0.006 W
Signals	0.009 W
Block RAM	0.004 W
I/O	< 0.001 W
XADC	0.002 W
Total dynamic	0.033 W
Device static	0.098 W
Total on-chip	0.131 W

Table 4.2 summarizes the vectorless Vivado estimate for the routed baseline design. The total estimated on-chip power is 0.131 W, of which 0.033 W is dynamic and 0.098 W is static. The dynamic component is mainly distributed among clocks, signals, slice logic, BRAM, and the XADC block.

The most important point is not the absolute value itself, but the low confidence level reported by Vivado. Since no SAIF file is used, the activity is based on default

tool assumptions rather than on the actual CoreMark workload. For this reason, the no-SAIF baseline report is used only as a reference point before introducing SAIF-annotated activity.

After the vectorless reference report, a second power report was generated using SAIF-annotated switching activity extracted from post-implementation simulation. In this case, the estimator no longer relies only on default activity propagation, but uses simulation-derived switching information associated with the executed CoreMark workload.

The SAIF-annotated report for the same routed baseline configuration is summarized in Table 4.3.

Table 4.3: Vivado SAIF-annotated power estimate for the routed baseline configuration.

(a) Power summary.

Quantity	**Value**
Device	xc7a100tcsg324-1
Design state	Routed
Total on-chip	0.134 W
Dynamic	0.035 W
Device static	0.098 W
Junction temp.	25.6 °C
Max ambient	84.4 °C
Effective θ_{JA}	4.6 °C/W
SAIF file	Yes
Design nets matched	73%
Confidence	High

(b) Dynamic breakdown.

Component	**Power**
Clocks	0.011 W
Slice logic	0.004 W
Signals	0.006 W
Block RAM	0.012 W
I/O	$<$ 0.001 W
XADC	0.002 W
Total dynamic	0.035 W
Device static	0.098 W
Total on-chip	0.134 W

Compared with the vectorless baseline estimate, the SAIF-annotated report gives a similar total on-chip power value, increasing from 0.131 W to 0.134 W. The dynamic component increases slightly from 0.033 W to 0.035 W, while the static component remains unchanged at 0.098 W.

The most relevant difference is the confidence level. In the vectorless report, Vivado assigns a low confidence level because the switching activity is inferred from default assumptions. In the SAIF-annotated report, the confidence level becomes high, since the power estimator uses switching activity extracted from post-implementation simulation. The report also indicates that 73% of the design nets were matched to the SAIF activity file.

The dynamic power distribution also changes. The estimated clock power remains 0.011 W, while the slice-logic and signal components decrease with respect to the vectorless estimate. Conversely, the Block RAM contribution increases from 0.004 W to 0.012 W, reflecting the workload-dependent activity associated with the simulated CoreMark execution and instruction-memory accesses.

Therefore, the SAIF-annotated baseline report should be interpreted as a more workload-aware tool estimate than the vectorless report. However, it is still a Vivado power estimate, not a physical measurement. Its role is to support the implementation workflow and to provide a more realistic baseline reference before comparing the modified clock-enable configurations.

The same SAIF-based procedure is then applied to the modified clock-enable configurations. These reports are used to check whether the estimated activity follows the expected trend when the CPU clock branch is forced off, permanently enabled, or driven dynamically by the sleep and wake-up logic. Nevertheless, the SAIF-annotated reports remain tool-based estimates and are therefore used only as intermediate evidence; the final validation of the power-saving effect is based on the physical measurements reported in Chapter 7..

4.5.4 Role of the Simulation-Based Power Workflow

The implementation and simulation-based power-analysis workflow is used as an intermediate validation step before the physical measurements. Its role is not to provide the final proof of the clock-gating benefit, but to verify that the design can be implemented correctly, that timing closure is preserved, and that the expected activity trends can be observed at tool level.

The continuous baseline configuration is the first reference point of this workflow. It confirms that the NEORV32/CoreMark system can be implemented with the selected internal memory configuration and can meet the 100 MHz timing constraint on the Arty A7-100T target. The corresponding vectorless power report provides an initial estimate of the routed design, but its confidence level is low because no simulation activity file is used.

The SAIF-annotated analysis improves this situation by introducing switching activity extracted from simulation. This makes the Vivado power report more representative of the executed workload than a purely vectorless estimate. However, the SAIF-based result is still a tool-generated estimate and remains dependent on the simulated activity window, the quality of the testbench, and the matching between the SAIF hierarchy and the implemented netlist.

For this reason, the Vivado reports are used only as supporting evidence. They help to check that the clock-gated and control designs behave consistently in the implementation flow, but they are not treated as the final experimental validation. The final evaluation of the proposed strategy is based on the physical measurements performed on the CW305 and Arty A7-100T platforms, where the active, sleep, and control configurations are compared using measured activity-related quantities.

4.6 CoreMark with Clock Gating Technique

After establishing a working NEORV32/CoreMark implementation and a repeatable Vivado implementation flow, the clock-gating technique was applied to the processor architecture. The objective of this step was to move from a continuously clocked processor to a design in which the CPU-side clock domain can be suppressed during controlled idle intervals.

The modified design is based on two complementary changes. At hardware level, the NEORV32 top-level is extended with a BUFGCE-based gated clock path. At software level, the CoreMark benchmark is instrumented so that execution alternates between active computation chunks and timer-driven sleep intervals. This structure makes it possible to analyze the effect of clock gating under a repeatable active/sleep execution pattern.

This part of the work has two roles. First, it defines the actual clock-gated architecture that will later be evaluated through physical measurements. Second, it provides a simulation-based power-analysis reference through Vivado reports, both with and without SAIF annotation. These reports are not treated as the final experimental evidence, but they are useful to verify that the clock-gating structure is correctly implemented and that the design remains compatible with the FPGA implementation flow.

4.6.1 Hardware and Software Implementation

At hardware level, the main modification consists in inserting a Xilinx `BUFGCE` primitive inside the `neorv32_top` module. The output of this primitive generates a dedicated gated clock signal, named `clk_cpu`. This clock is used for the CPU-side part of the system, while the global clock remains available for the logic that must continue operating during sleep intervals.

This separation is essential. If the complete SoC were clock-gated, the processor could enter a sleep state and never wake up, because the timer and interrupt infrastructure would also be stopped. For this reason, the CLINT timer, the UART interface, and the measurement-related logic are kept on the global clock. The gated clock is instead used for the logic that is intended to reduce switching activity during the sleep intervals.

The BUFGCE clock-enable signal is generated from the processor sleep state and from the wake-up sources required to resume execution. The implemented condition is:

```
-- Clock ON when CPU is not sleeping, during reset, or on wake-up
   events
ce_cpu <= (not rstn_sys) or
          (not cpu_sleep(0)) or
          mti(0) or
          msi(0) or
          irq_mei_i or
          or_reduce_f(cpu_firq);
```

The term `not rstn_sys` keeps the CPU clock active during reset, avoiding the risk of freezing the processor domain while the system is being initialized. The term `not cpu_sleep(0)` keeps the clock enabled during normal execution. The remaining terms re-enable the clock when a wake-up source becomes pending, including timer, software, external, and fast interrupt events.

On the software side, the CoreMark benchmark is modified to generate a controlled active/sleep pattern. Instead of executing the benchmark as a single continuous block, the workload is divided into chunks. During each active chunk, the processor executes the CoreMark workload. After the chunk, the software programs the machine timer compare register and then executes `neorv32_cpu_sleep()`, which relies on the RISC-V `WFI` instruction.

When the programmed timer interval expires, the machine timer interrupt becomes pending. This event asserts the wake-up path, the BUFGCE enable becomes active again, and the CPU clock is restored. The processor then resumes execution and proceeds with the next CoreMark chunk. In this way, the software creates repeatable active and sleep windows, while the hardware determines whether the CPU clock is actually propagated or suppressed during the sleep interval.

For the purpose of comparison, three clock-path configurations are considered during the implementation and power-analysis phase:

- **CE forced low**: the BUFGCE enable is forced to zero. This case is used only as a lower switching-activity reference for the gated CPU clock branch.
- **CE forced high**: the BUFGCE enable is forced to one. In this case, the BUFGCE path is present, but the CPU clock is continuously propagated. This is the `CE = 1` control configuration.
- **Dynamic CE**: the BUFGCE enable is driven by the actual sleep and wake-up logic. This is the intended clock-gated implementation evaluated in the experimental campaign.

The forced cases are not meant to represent the final runtime behavior of the processor. They are used to understand the effect of the inserted clock path and to provide reference points for the Vivado power reports. The dynamic case is the actual implementation used to evaluate the effect of combining `WFI`-based sleep intervals with real CPU clock suppression.

4.6.2 Resource Utilization

Before evaluating the power reports, the implemented resource utilization of the clock-gated design was checked. The objective of this step is not to quantify the power saving itself, but to verify that the architectural modification remains compatible with the target FPGA implementation flow.

The insertion of the BUFGCE-based clock path adds a small amount of clock-control logic and modifies the clocking structure of the processor domain. In addition, the design includes the infrastructure required for the experimental campaign, such as GPIO-based triggering, UART communication, timer-based wake-up, and the XADC monitoring logic used in the Arty A7-100T measurements. Therefore, the implemented utilization is expected to differ from the continuous baseline configuration.

Figure 4.4 reports the post-implementation device usage for the clock-gated NEORV32/CoreMark configuration. The purpose of this check is to confirm that the design can still be synthesized, placed, routed, and timing-checked after the clock-gating modification. The power-saving effect is not inferred from this figure, but from the subsequent power reports and physical measurements.

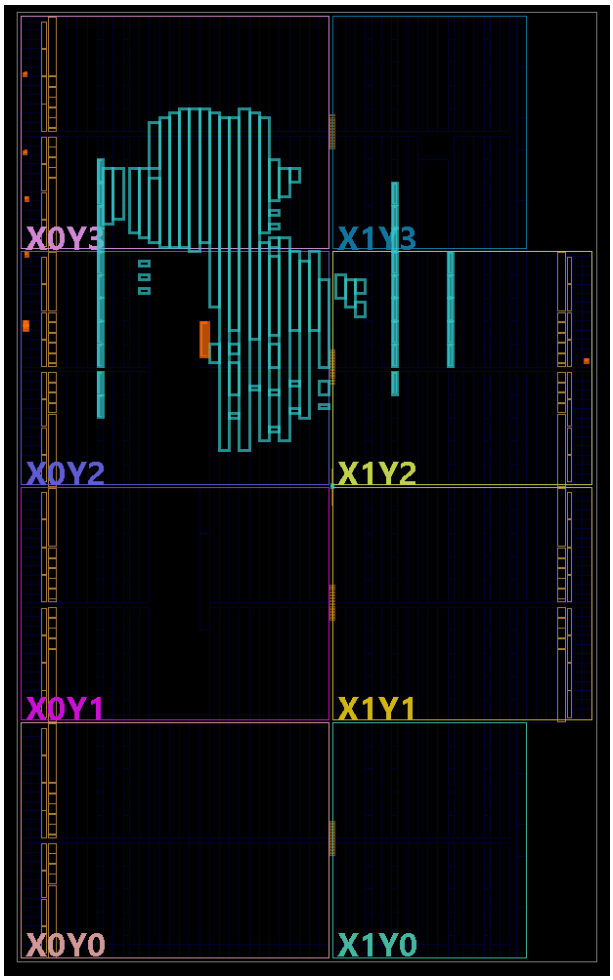


Figure 4.4: Post-implementation device usage for the clock-gated NEORV32/CoreMark configuration on the Arty A7-100T platform.

4.6.3 Phase 1: Vectorless Power Analysis Without SAIF

The first vectorless power-analysis phase was performed during the implementation bring-up of the BUFGCE-based clock path. At this stage, the objective was not yet to reproduce the final physical measurement campaign described in the following chapters. Instead, the purpose was to verify that the inserted clock-gating structure was controllable and that its effect could be observed at least qualitatively in the Vivado power reports.

For this reason, three clock-enable conditions were considered:

- `CE = 0`: the BUFGCE enable is forced low. In this condition, the CPU-side gated clock is suppressed. This case is used as a lower switching-activity reference for the gated clock branch.
- `CE = 1`: the BUFGCE enable is forced high. In this condition, the BUFGCE path is present in the design, but the CPU clock is continuously propagated. This case is used as an always-clocked reference for the modified clock path.
- `CE = free`: the BUFGCE enable is driven by the implemented sleep and wake-up logic. This is the dynamic clock-enable case, where the enable is not forced to a constant value.

These configurations do not correspond directly to the final physical measurement campaign. Their role is more limited: they provide lower, upper, and dynamic reference points for the clock-gating path during the implementation phase.

Since this first analysis was performed without SAIF annotation, Vivado relied on vectorless/default switching activity. Therefore, the reported values should be interpreted only as preliminary estimates. In particular, the tool does not know the actual active/sleep duty cycle of the CoreMark execution, and it cannot accurately represent how long the CPU clock is enabled or disabled during runtime.

Table 4.4 summarizes the vectorless Vivado power estimates obtained for the three clock-enable configurations.

Table 4.4: Preliminary vectorless Vivado power estimates for the clock-enable configurations.

Quantity	`CE = 0`	`CE = 1`	`CE = free`
Device	xc7a100tcsg324-1	xc7a100tcsg324-1	xc7a100tcsg324-1
Design state	Routed	Routed	Routed
Total on-chip power	0.096 W	0.140 W	0.139 W
Dynamic power	0.011 W	0.042 W	0.040 W
Device static power	0.085 W	0.098 W	0.098 W
Junction temperature	25.4 °C	25.6 °C	25.6 °C
Max ambient temperature	84.6 °C	84.4 °C	84.4 °C
Effective θ_{JA}	4.6 °C/W	4.6 °C/W	4.6 °C/W
Simulation activity file	–	–	–
Overall confidence level	Low	Low	Low

A more detailed dynamic power breakdown is reported in Table 4.5. When `CE = 0`, the estimated dynamic power decreases to 0.011 W. When `CE = 1`, the clock is continuously propagated and the estimated dynamic power increases to 0.042 W. In the `CE = free` case, the vectorless estimate remains close to the `CE = 1` case, with 0.040 W of dynamic power.

Table 4.5: Dynamic power breakdown for the clock-enable configurations without SAIF annotation.

Component	`CE = 0`	`CE = 1`	`CE = free`
Clocks	0.005 W	0.016 W	0.014 W
Slice logic	0.002 W	0.008 W	0.008 W
Signals	0.002 W	0.012 W	0.012 W
Block RAM	0.000 W	0.004 W	0.004 W
I/O	< 0.001 W	< 0.001 W	< 0.001 W
XADC	0.002 W	0.002 W	0.002 W
Total dynamic	0.011 W	0.042 W	0.040 W
Device static	0.085 W	0.098 W	0.098 W
Total on-chip	0.096 W	0.140 W	0.139 W

The comparison confirms that forcing the BUFGCE enable low reduces the estimated clock, signal, logic, and BRAM dynamic components. This is consistent with the expected behavior of the inserted gated clock path: when the enable is forced low, a large portion of the CPU-side switching activity is suppressed; when it is forced high, the clocked logic remains active.

The `CE = free` case is close to the `CE = 1` case in the vectorless reports. This is expected because no SAIF file is used. Vivado does not know the actual runtime activity of the dynamic clock-enable signal and therefore cannot accurately represent the active/sleep duty cycle of the CoreMark execution. As a result, the vectorless `CE = free` report is useful only as an implementation sanity check, not as a quantitative evaluation of the final clock-gating benefit.

Overall, this phase confirms that the BUFGCE-based clock path is controllable and visible in the Vivado power reports. However, the values remain preliminary because the switching activity is not workload-specific. For this reason, the following phase uses SAIF annotation to include activity extracted from simulation.

4.6.4 Phase 2: SAIF-Annotated Power Analysis

The second power-analysis phase uses switching activity extracted from simulation. Unlike the vectorless reports discussed in the previous section, SAIF-annotated reports include activity information generated during the simulated CoreMark execution. This makes the power estimate more representative of the actual workload applied to the implemented design.

The purpose of this phase is to improve the quality of the Vivado power estimate before moving to physical measurements. However, the SAIF-annotated reports are still tool-based estimates. They depend on the simulated time window, on the quality of the testbench, and on the matching between the SAIF hierarchy and the implemented netlist. For this reason, they are used as intermediate validation evidence, not as the final proof of the power-saving effect.

The same three clock-enable configurations considered in the vectorless phase are used:

- `CE = 0`: the clock enable is forced low. This case represents the lower switching-activity reference for the CPU-side gated clock path.

- `CE = 1`: the clock enable is forced high. This case represents the always-clocked reference with the BUFGCE path present.
- `CE = free`: the clock enable is driven by the implemented sleep and wake-up logic. This is the dynamic clock-gated case.

The `CE = 0` and `CE = 1` cases are forced reference conditions, while the `CE = free` case corresponds to the intended dynamic behavior of the clock-gating logic. The comparison between these three reports is used to verify whether the SAIF-annotated Vivado estimate follows the expected trend: lower activity when the CPU clock is suppressed, higher activity when it is continuously propagated, and an intermediate or workload-dependent behavior when the enable is controlled dynamically.

Table 4.6: SAIF-annotated Vivado power estimates for the clock-enable configurations.

Quantity	`CE = 0`	`CE = 1`	`CE = free`
Device	xc7a100tcsg324-1	xc7a100tcsg324-1	xc7a100tcsg324-1
Design state	Routed	Routed	Routed
Total on-chip power	0.093 W	0.134 W	0.132 W
Dynamic power	0.008 W	0.036 W	0.033 W
Device static power	0.085 W	0.098 W	0.098 W
Junction temperature	25.4 °C	25.6 °C	25.6 °C
Max ambient temperature	84.6 °C	84.4 °C	84.4 °C
Effective θ_{JA}	4.6 °C/W	4.6 °C/W	4.6 °C/W
Simulation activity file	SAIF	SAIF	SAIF
Design nets matched	72%	72%	72%
Overall confidence level	Low	High	High

Table 4.7: SAIF-annotated dynamic power breakdown for the clock-enable configurations.

Component	`CE = 0`	`CE = 1`	`CE = free`
Clocks	0.005 W	0.015 W	0.013 W
Slice logic	$<$ 0.001 W	0.004 W	0.004 W
Signals	$<$ 0.001 W	0.005 W	0.005 W
Block RAM	0.000 W	0.009 W	0.009 W
I/O	$<$ 0.001 W	$<$ 0.001 W	$<$ 0.001 W
XADC	0.002 W	0.002 W	0.002 W
Total dynamic	0.008 W	0.036 W	0.033 W
Device static	0.085 W	0.098 W	0.098 W
Total on-chip	0.093 W	0.134 W	0.132 W

Tables 4.6 and 4.7 summarize the SAIF-annotated Vivado power estimates for the three clock-enable configurations. Compared with the vectorless reports, these estimates

include switching activity extracted from post-implementation simulation and are therefore more representative of the simulated CoreMark execution.

The expected trend is visible in the SAIF-annotated reports. When the clock enable is forced low, the `CE = 0` configuration shows the lowest estimated dynamic power, equal to 0.008 W. This is consistent with the suppression of the CPU-side gated clock branch. However, this report still has a low confidence level because Vivado reports insufficient clock-node activity coverage. Therefore, the `CE = 0` case should be interpreted only as a lower switching-activity reference, not as a high-confidence runtime operating point.

The `CE = 1` configuration provides the always-clocked reference with the BUFGCE path present. In this case, the estimated total on-chip power is 0.134 W, with 0.036 W of dynamic power. The clock component is 0.015 W, confirming that the CPU clock is continuously propagated through the modified clock path.

The `CE = free` configuration, where the clock enable is driven by the implemented sleep and wake-up logic, gives an estimated total on-chip power of 0.132 W, with 0.033 W of dynamic power. This value is slightly lower than the `CE = 1` case, mainly due to the lower estimated clock contribution, which decreases from 0.015 W to 0.013 W. This indicates that the SAIF-annotated analysis captures some reduction associated with the dynamic clock-enable behavior.

Nevertheless, the `CE = free` result remains close to the `CE = 1` result. This confirms that the SAIF-annotated Vivado report should not be treated as final experimental proof of the clock-gating benefit. The estimate depends on the simulated time window, the activity represented in the SAIF file, and the hierarchy matching between the simulation and the implemented netlist. In the present reports, the matched design nets are approximately 72% for all three modified configurations.

Overall, the SAIF-annotated analysis improves the quality of the tool-based estimate with respect to the vectorless phase and confirms that the inserted clock-gating path is controllable. The forced-low case provides the lower activity reference, the forced-high case provides the always-clocked reference, and the dynamic case lies close to the always-clocked case but with a slightly reduced dynamic estimate. For this reason, these reports are used as intermediate implementation-level evidence, while the final validation of the power-saving effect is based on the physical measurements reported in Chapter 7.

Chapter 5

Hardware Platforms and Software Environment

This chapter describes the hardware platforms and software tools used throughout the experimental work. Two Artix-7 based platforms were considered. The CW305 board was used for oscilloscope-based power-trace measurements through its shunt measurement path, while the Arty A7-100T platform was used for an integrated XADC-based current-monitoring flow.

The chapter also introduces the main software tools used during the project. Vivado was used for synthesis, implementation, timing verification, and power estimation. GHDL and GTKWave were used during the RTL validation phase to simulate the VHDL design and inspect relevant internal signals.

5.1 CW305 Overview

The NewAE NAE-CW305 is an FPGA target board designed for side-channel analysis and power-measurement experiments. The board can host an Artix-7 FPGA and provides dedicated measurement access to the FPGA core supply path, making it suitable for observing activity variations caused by hardware modifications such as clock gating [19].

In this thesis, the CW305 was used as the first physical validation platform for the NEORV32/CoreMark implementation. The FPGA was programmed with the NEORV32 design, while an oscilloscope was used to acquire both the GPIO-based trigger signal and the shunt-related activity waveform. This made it possible to correlate the benchmark execution window with the activity observed on the FPGA core supply path.

The main components of the CW305 setup used in this work are:

- **USB interface**: used to power and communicate with the board, as well as to configure the FPGA from the host PC.
- **Artix-7 FPGA**: used as the target device for the NEORV32 processor and the CoreMark workload.
- **PLL clocking system**: used to provide a controlled clock signal to the FPGA design.
- **Trigger output**: used to export the GPIO-based trigger generated by the NEORV32 software and observed on the oscilloscope.

- **VCC-INT shunt measurement path**: used to observe activity-related variations on the FPGA core supply rail.
- **Amplified shunt output**: connected to the oscilloscope to acquire the power-related waveform during benchmark execution.

The relevance of the CW305 in this work is therefore not its use as a generic cryptographic evaluation board, but its measurement infrastructure. In particular, the combination of FPGA programmability, controlled clocking, trigger access, and shunt-based power observation makes it suitable for validating whether the proposed clock-gating strategy produces visible changes in the measured activity.

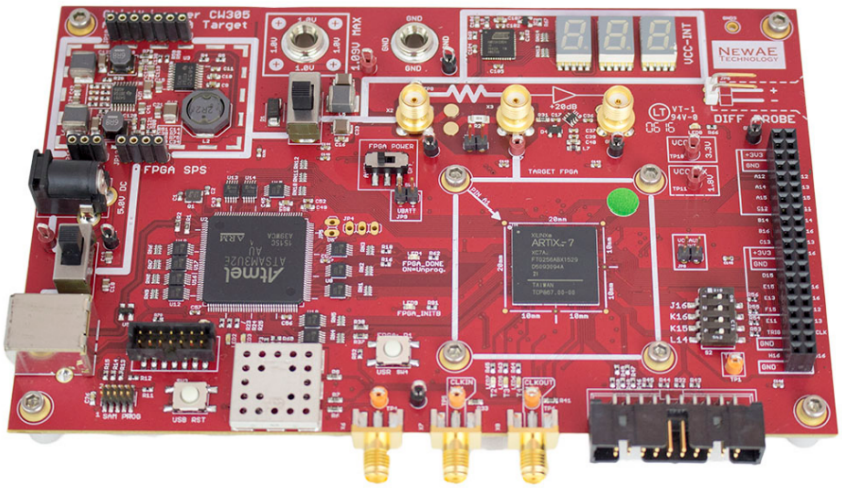


Figure 5.1: CW305 board.

5.1.1 Hardware Setup

For the purpose of this thesis, the CW305 was used as an FPGA target connected to an external oscilloscope. The NEORV32 system implemented on the FPGA generated a GPIO-based trigger signal to mark the benchmark execution window, while the shunt-related output of the board was used to observe activity variations on the FPGA core supply path. [19]

The oscilloscope acquisition was based on two channels: one channel was connected to the trigger signal, and the other to the amplified shunt-related output. This allowed the benchmark timing and the power-related waveform to be observed simultaneously. The detailed acquisition procedure and the complete connection scheme are described later in Chapter 6.

This section therefore only introduces the hardware role of the CW305 platform, while the experimental measurement methodology is discussed in the dedicated setup chapter.

5.1.2 USB Interface

The CW305 includes an on-board USB interface used for communication with the host PC, FPGA configuration, and board-level control. Through this interface, the user can

program the FPGA, configure the board, and access control registers exposed by the target platform.

At a higher level, the CW305 software interface supports SimpleSerial-style commands, such as *simpleserial_read()* and *simpleserial_write()*, when compatible target firmware is used. These functions are useful in side-channel analysis workflows where the host needs to send input data to the FPGA design and read back output data.

For lower-level communication, the CW305 also allows direct register access. This means that the user can read and write board registers without relying only on the serial *read()*/*write()* mechanism provided by the SimpleSerial class. This feature is useful for configuring board resources, clock settings, and measurement-related options.

Another useful feature is the possibility of disabling selected USB-related clocks during sensitive measurements. This can reduce additional switching noise coupled into the power measurement path. Such board-level control can be performed through the CW305 Python API [20].

In this thesis, the USB interface was mainly used for FPGA configuration and board setup. The benchmark execution was controlled by the NEORV32 design implemented on the FPGA, while the power-related waveform was acquired externally with the oscilloscope.

5.1.3 PLL

The CW305 implements a CDCE906 programmable PLL for generating the FPGA clock. The PLL can be configured through the USB interface and can generate output frequencies in the approximate range from 5 MHz to 160 MHz. It provides three clock outputs: two are routed to FPGA pins, while one can be routed to an SMA connector for external observation or synchronization purposes [20].

In the reference CW305 clocking scheme, PLL output 1 is commonly used as the FPGA clock source, as shown in Fig. 5.2. In this thesis, the clocking infrastructure of the board was relevant because the NEORV32 design had to be executed at a controlled and repeatable system frequency during the benchmark measurements.

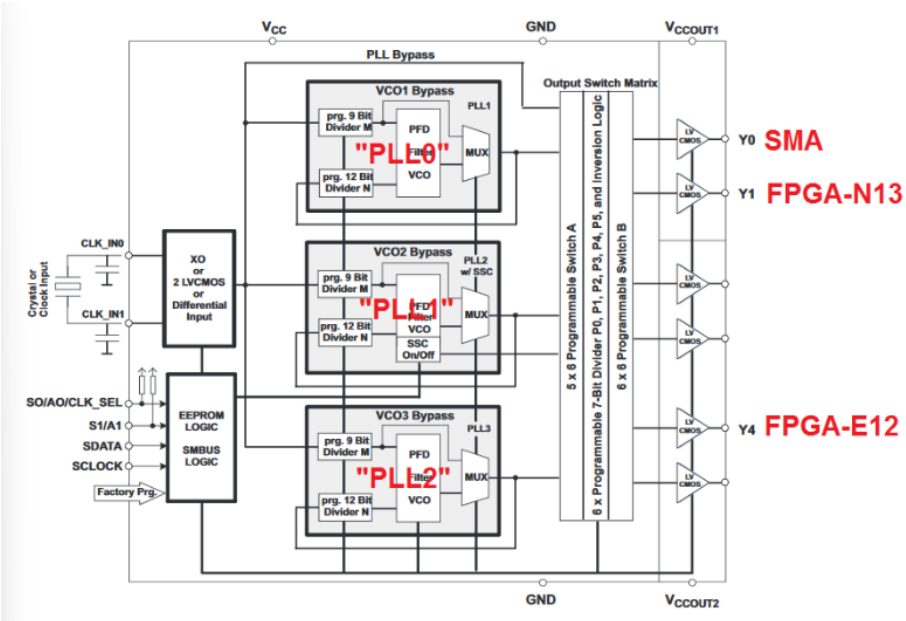


Figure 5.2: CW305 PLL architecture.

PLL1 and PLL2 are connected to FPGA pins N13 and E12, respectively. The SMA connector X6 can be connected to one of the PLL outputs, allowing an external phase-matched clock signal to be observed when required [20].

5.1.4 Power Supply

The CW305 board includes three switching regulators for the FPGA supply rails. Two of them provide fixed voltages for the auxiliary and I/O domains, namely VCC-AUX (1.8 V) and VCC-IO (3.3 V). The remaining regulator supplies the FPGA core rail, VCC-INT, and can be programmed through the USB interface. The programmable regulator can output approximately 0.65 V to 1.5 V, although the software interface restricts the usable range to about 0.8 V–1.10 V to avoid damaging the FPGA. The VCC-INT regulator is capable of supplying several amperes to the FPGA core [20].

The VCC-INT rail is the most relevant supply rail for this thesis, because it powers the FPGA core logic where the NEORV32 processor is implemented. The shunt-based measurement path used in the CW305 campaign is placed on this supply path, allowing activity variations of the FPGA core to be observed during benchmark execution.

Power can be supplied to the CW305 either through the USB connector or through an external 5 V DC input. For larger FPGA designs, an external supply may be preferred, since the USB current capability can become a limiting factor [20].

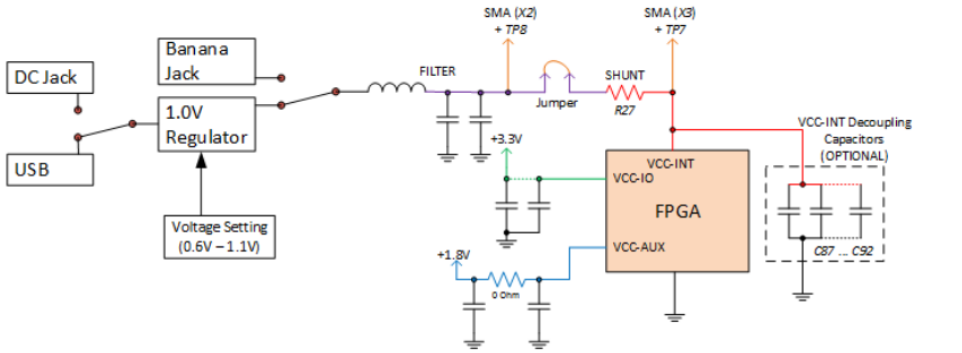


Figure 5.3: CW305 power supply routing.

5.1.5 Shunt Measurement

A resistive shunt is placed between the VCC-INT supply rail and the FPGA core supply path. This makes it possible to observe current-related variations associated with the activity of the FPGA core logic. Since the NEORV32 processor is implemented inside the FPGA fabric, this measurement path is directly relevant for comparing active and idle processor behavior.

The CW305 provides several access points for observing the shunt signal. SMA connectors give access to both sides of the shunt: the high side is available on X2, while the low side is available on X3. In addition, the board provides a low-noise amplified version of the low-side signal on X4, with a nominal gain of 20 dB. The shunt voltage can also be measured through differential probe headers: JP7 is located close to the shunt resistor, while JP6 is placed on the upper-right side of the PCB [20].

In this thesis, the amplified shunt-related output was used as the main power-related observation signal during the CW305 measurements. This signal was acquired with an oscilloscope and correlated with the GPIO-based trigger generated by the NEORV32 software. In this way, changes in the measured waveform could be interpreted in relation to active and idle benchmark intervals.

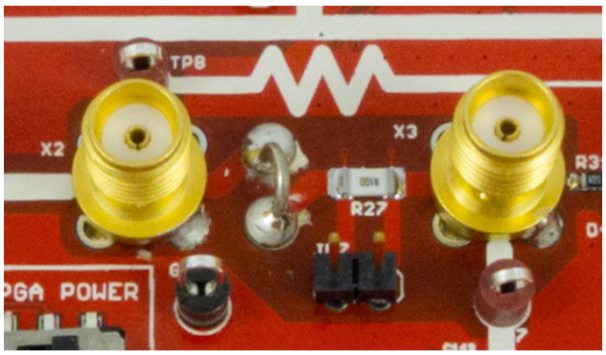


Figure 5.4: CW305 shunt resistor.

Note: the schematic representation and the physical PCB layout show the power path in opposite directions. On the PCB, the power path is observed from left to right, whereas in the schematic representation it is shown from right to left.

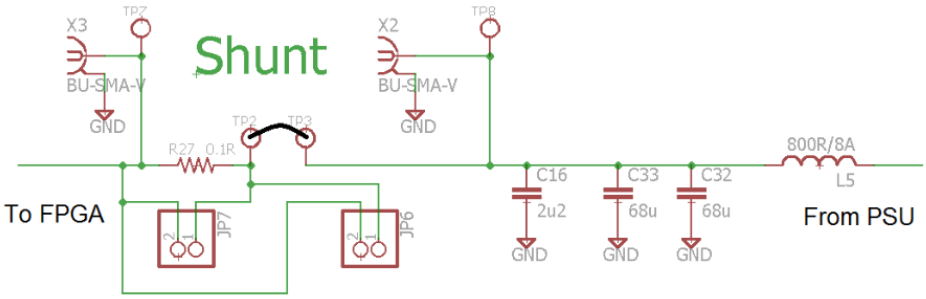


Figure 5.5: CW305 shunt resistor schematic.

5.1.6 CW305 to Oscilloscope

The CW305 can be connected to standard external instrumentation, such as an oscilloscope, when a ChipWhisperer capture platform is not used. This configuration is useful when the objective is to directly observe the trigger signal and the shunt-related waveform during FPGA execution, as done in this thesis [19].

The typical connection procedure is the following:

1. Ensure that the CW305 board is powered off before making the measurement connections, in order to avoid hot-plugging issues.

2. Connect an SMA-to-BNC cable from the oscilloscope input to the amplified shunt output of the CW305 board.

3. Connect an oscilloscope probe to the trigger test point, labelled `TRIG`, so that the benchmark execution window can be observed.

4. Power on the CW305 board, or connect the external power supply if it was disconnected.

5. Configure the oscilloscope channel connected to the shunt output according to the desired acquisition mode. In this work, the shunt-related waveform was used as a comparative activity indicator rather than as an absolute DC power measurement.

In the experimental setup adopted in this thesis, one oscilloscope channel was used for the GPIO-based trigger generated by the NEORV32 software, while another channel was used for the amplified shunt-related signal. This allowed the benchmark execution window to be correlated with the observed FPGA core activity.

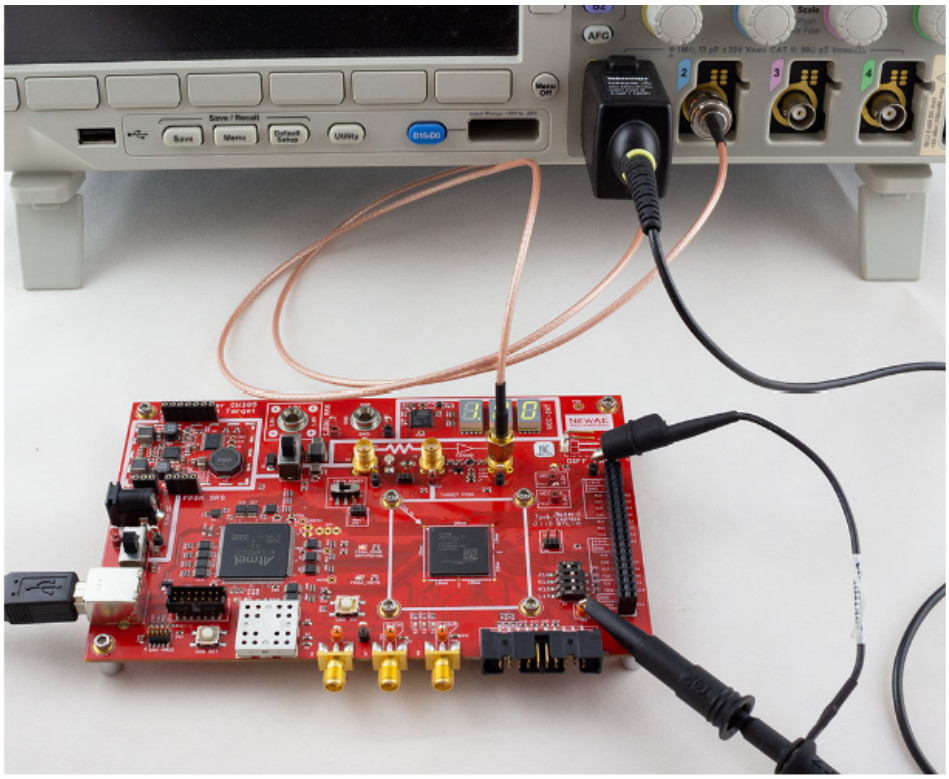


Figure 5.6: CW305 connection to an oscilloscope.

5.2 Arty A7-100T Platform

The Arty A7-100T is a ready-to-use development board based on the Xilinx Artix-7™ Field Programmable Gate Array (FPGA). The board is commonly used for soft-processor systems and is compatible with the Vivado Design Suite. Although it is often associated with MicroBlaze-based designs, its FPGA fabric can also be used to implement custom VHDL systems, such as the NEORV32-based architecture used in this thesis [21].

The flexibility of the Arty A7 platform comes from the reconfigurable nature of the FPGA. Unlike a fixed single-board computer, the board is not limited to a predefined set of processing peripherals. The designer can instantiate the required processor, memories, interfaces, and custom hardware modules directly in the FPGA fabric. This makes the platform suitable for experimenting with architectural modifications such as the BUFGCE-based clock-gating strategy investigated in this work.

In this thesis, the Arty A7-100T was used as the second physical validation platform. The NEORV32 processor, the CoreMark program image, the GPIO trigger interface, the UART output path, and the XADC-based current-monitoring module were implemented on the FPGA. This allowed the board to execute the benchmark and collect current-related samples internally, providing a more automated measurement flow with respect to the oscilloscope-based CW305 setup.

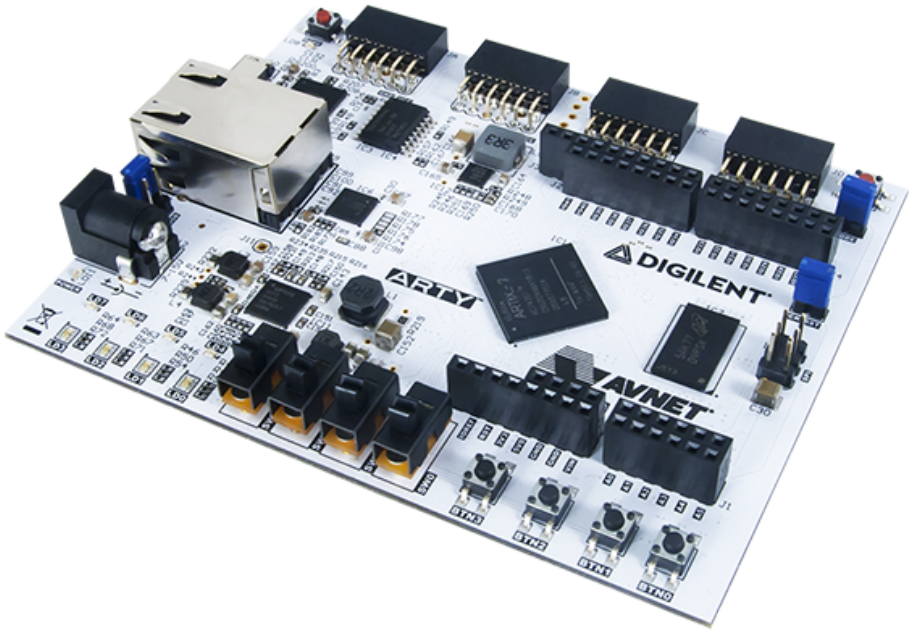


Figure 5.7: Arty A7 board.

The Arty A7 is fully compatible with the Vivado Design Suite, which was used in this work for synthesis, implementation, timing analysis, bitstream generation, and power estimation.

5.2.1 Designing with the Arty A7

The flexibility of the Arty A7 platform comes from the reconfigurable nature of its FPGA. An FPGA can be configured as a custom software-defined System-on-Chip (SoC), where the processor, memories, peripherals, and custom logic are implemented directly in the programmable fabric.

A common way to build soft-SoC designs on the Arty A7 is through Vivado IP Integrator (Vivado IPI), shown in Fig. 5.8. In this environment, the designer can instantiate pre-built IP blocks and connect them graphically to create a complete processing system. Typical peripherals include timers, UART, SPI, and IIC controllers. Custom hardware

blocks can also be added by writing them in a Hardware Description Language (HDL), such as Verilog or VHDL.

The Arty A7 is often used with MicroBlaze, a 32-bit soft processor core designed for Xilinx FPGAs. A MicroBlaze-based SoC can typically operate around 100 MHz, and higher operating frequencies may be possible depending on the design and timing constraints. The board also provides external memory resources, including non-volatile flash and DDR3L memory, which can support larger embedded applications [21].

In this thesis, however, the Arty A7-100T is not used with MicroBlaze. Instead, the FPGA fabric is used to implement the NEORV32 RISC-V processor and the custom VHDL logic required for the measurement campaign. This includes the GPIO trigger interface, UART output, internal memories, and the XADC-based current-monitoring module. Therefore, the same flexibility that normally enables MicroBlaze-based soft SoCs is exploited here to evaluate a custom RISC-V architecture with clock-gating support.

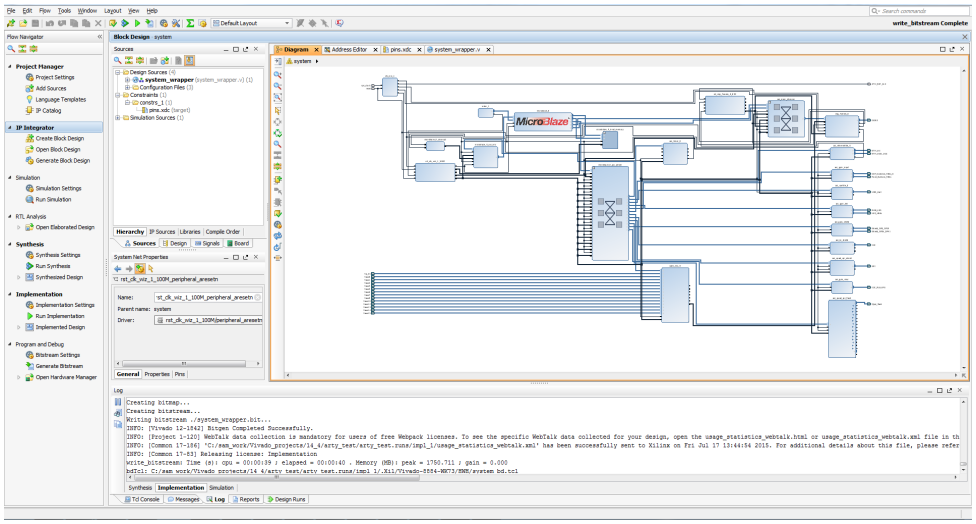


Figure 5.8: Vivado IP Integrator environment commonly used for Arty-based soft-SoC designs.

5.2.2 Power Supplies

The Arty A7 requires a 5-volt power source to operate. This power source can be supplied via the Digilent USB-JTAG port (J10) or derived from a 7 to 15-volt DC power supply connected to the power jack (J13) or Pin 8 of Header J7. A power-good LED (LD11), driven by the 3.3V output (VCC3V3) of the DA9062 regulator, indicates that the board is receiving power and that the onboard supplies are functioning as expected.

The USB port can deliver sufficient power for the vast majority of designs. However, a few demanding applications might require more power; in such cases, the solution is to use an external power supply or a battery pack [21].

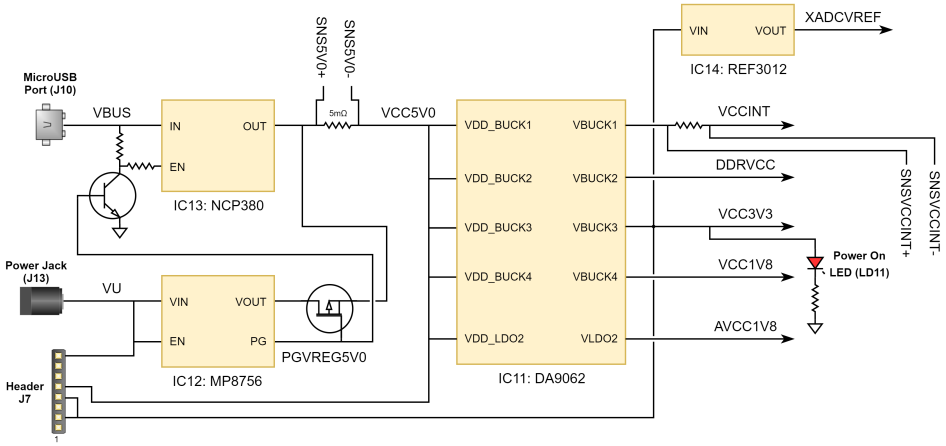


Figure 5.9: Arty A7 Power Circuit

An external power supply can be connected to the power jack J13 5.9. The supply must use a coaxial, center-positive 2.1 mm internal-diameter plug and provide a voltage between 7 and 15 Volts DC. The supply should provide a minimum current of 1A [22].

5.2.3 FPGA Configuration

After power-on, the Artix-7 FPGA must be configured (or programmed) before it can perform any functions. You can configure the FPGA in one of two ways:

1. A PC can use the Digilent USB-JTAG circuitry (port J10) to program the FPGA at any time while the power is on.

2. A file stored in the non-volatile serial (SPI) flash device can be transferred to the FPGA using the SPI port.

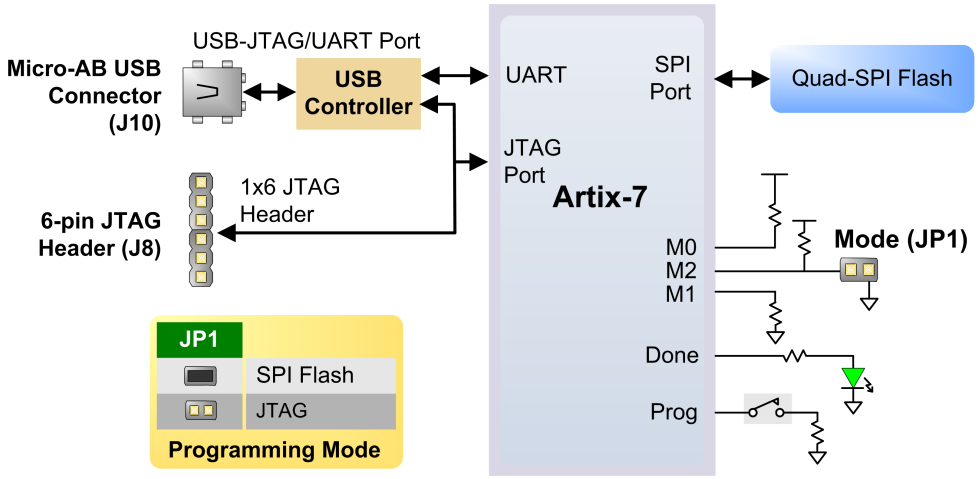


Figure 5.10: Arty A7 FPGA Configuration Options

Figure 5.10 shows the different options available for configuring the FPGA. An on-board "mode" jumper (JP1) selects whether the FPGA will be programmed by the Quad-SPI flash upon power-up.

The FPGA configuration data is stored in files called bitstreams, which have the .bit file extension. The ISE or Vivado software from Xilinx can create bitstreams from VHDL,

Verilog®, or schematic-based source files. Bitstreams are stored in volatile memory cells within the FPGA. This data defines the FPGA's logic functions and circuit connections; it remains valid until it is erased by removing board power, by pressing the reset button attached to the PROG input, or by writing a new configuration file using the JTAG port [21].

5.2.4 Memory

The Arty A7 features two external memories: a 256 MB DDR3L SDRAM and a 128 Mb (16 MB) non-volatile serial Flash device. The DDR3L module is connected to the FPGA using an industry-standard interface, while the serial Flash is connected via a dedicated quad-mode (x4) SPI bus [22].

5.3 Software Environment

In the modern era, the use of simulation software is crucial to determine if a project can be realized before investing in hardware setup and physical implementation. In this thesis, three principal software tools have been used: Vivado, GTKWave, and GHDL.

5.3.1 Vivado

The AMD Vivado Design Suite is engineered to enhance productivity. This tool suite is architected to increase overall efficiency in designing, integrating, and implementing systems. Within Vivado, it is possible to accelerate design implementation using place and route tools that analytically optimize for multiple concurrent design metrics, such as timing, congestion, total wire-length, utilization, and power.

All Vivado Design Suite tools are built with a native Tool Command Language (Tcl) interface. All commands and options available in the Vivado Integrated Design Environment (IDE), which is the graphical user interface (GUI) for the Vivado Design Suite, are accessible through Tcl. The Vivado Design Suite also provides powerful access to design data for reporting and configuration, as well as for executing tool commands and options.

Interaction with the Vivado Design Suite can occur through:

- GUI-based commands within the Vivado IDE;
- Tcl commands entered in the Tcl Console of the Vivado IDE, in the Vivado Design Suite Tcl shell outside the IDE, or saved to a Tcl script file;
- A combination of GUI-based and Tcl commands.

A Tcl script can contain commands covering the entire design synthesis and implementation flow, including all necessary reports generated for design analysis at any point in the process.

The Vivado IDE provides new users with an intuitive interface while offering advanced users the power they require. All tools and settings are written in native Tcl. Analysis can be performed and constraints assigned throughout the design process; for example, the tools can provide timing or power estimations after synthesis, placement, or routing. Because the database is accessible through Tcl, changes to constraints, design configuration, or tool settings can be made in real time, often without forcing re-implementation.

The Vivado IDE introduces the concept of opening designs in memory. Opening a design effectively loads the design netlist at that particular stage of the design flow, assigns the constraints, and applies the design to the target device. This allows the user to visualize and interact with the design at each stage. The Vivado IDE enables the opening of designs after Register-Transfer Level (RTL) elaboration, synthesis, and implementation. Users can modify constraints, logic, or device configurations, and analyze implementation results. Design checkpoints can also be used to save the current state of any design. A design checkpoint is a snapshot of the design at any stage of the process that includes the netlist, constraints, and implementation results. Vivado automatically creates design checkpoints at each stage of the flow, which can be subsequently opened and analyzed [16].

5.3.2 GTKWave

The GTKWave visualization tool primarily consists of a menu bar, an embedded Signal Search Tree (SST) expander, various control buttons, a time status section, and dedicated frames for signals and wave values.

Both the main window size and position, as well as the current viewer state, can be saved between sessions. This includes information regarding which signals are visible, specific attributes such as alignment and inversion, marker positions, and active pattern marking. A frame labeled "Signals" is located on the middle left of the interface. Within this frame, signal names can be left- or right-aligned, and the user can define the number of hierarchy levels to be displayed. To the right of the signal section is the "Waves" frame. The top line serves as a timescale, while all other lines are used to render trace value data relative to that timescale. Analog traces of varying heights can also be displayed; these can be dynamically resized to facilitate viewing [23]. However, their size is limited to integer multiples of the height of a single digital trace (Figure 5.11).

The embedded SST section is located to the left of the signal section.

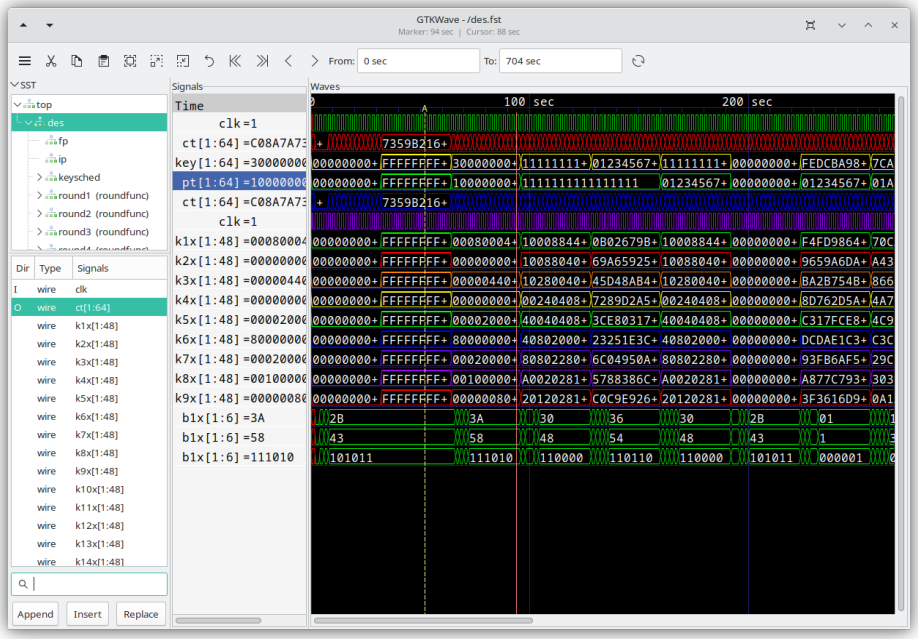


Figure 5.11: GTKWave main window

5.3.3 GHDL

GHDL is shorthand for "G Hardware Design Language" (currently, the "G" has no specific meaning). It is a VHDL analyzer, compiler, and simulator capable of executing nearly any VHDL program. GHDL is not a synthesis tool; therefore, it cannot be used to create a netlist directly.

Unlike many other simulators, GHDL acts as a compiler: it translates VHDL files directly into machine code without relying on an intermediary language such as C or C++. Consequently, the compiled code typically offers faster execution and shorter analysis times compared to simulators using intermediary languages.

GHDL supports multiple back-ends (code generators), including GCC, LLVM, or a built-in mcode generator (x86/i386 only). It is compatible with GNU/Linux, Windows®, and macOS® on both x86 and x86_64 architectures.

GHDL does not include a built-in graphical viewer for signal waveforms. Nevertheless, users can verify design behavior using a testbench. Furthermore, GHDL can produce output files in GHW, VCD, or FST formats, which can be opened and analyzed using external waveform viewers such as GTKWave.

GHDL aims to implement the VHDL standard as defined by IEEE 1076. It currently supports the 1987, 1993, and 2002 revisions and provides partial support for the 2008 revision. Additionally, PSL (Property Specification Language) is partially supported [24].

Chapter 6

Experimental Setup and Methodology

This chapter describes the experimental setup and measurement methodology used to evaluate the proposed clock-gating strategy on FPGA hardware. The objective is to define the tested configurations, the acquisition platforms, and the processing flow used to obtain the results discussed in the following chapter.

The experimental campaign was organized around two Artix-7 based platforms. The CW305 board was used for oscilloscope-based measurements of the shunt-related activity signal, providing a direct observation of the benchmark execution window and of the corresponding power-related waveform. The Arty A7-100T platform was then used to implement an integrated XADC-based acquisition flow, in which the current-sense signal is sampled inside the FPGA and exported through UART for MATLAB post-processing.

The same conceptual comparison is used across both platforms. The continuous baseline configuration provides the reference behavior without explicit idle intervals. The sleep-aware configuration introduces WFI-based idle windows without effective clock suppression. The CE = 1 control configuration keeps the BUFGCE-based clock path in the design but forces its enable signal permanently high. Finally, the clock-gated configuration combines WFI-based sleep intervals with actual CPU clock suppression.

The chapter is structured as follows. First, the CW305 setup is described, including the trigger generation, the shunt-based measurement path, and the oscilloscope acquisition methodology. Then, the tested configurations are defined. Finally, the Arty A7-100T measurement setup is introduced, with particular focus on the XADC-based acquisition flow and on the limitations of the measurement chain. The numerical results and the comparative analysis are presented separately in Chapter 7.

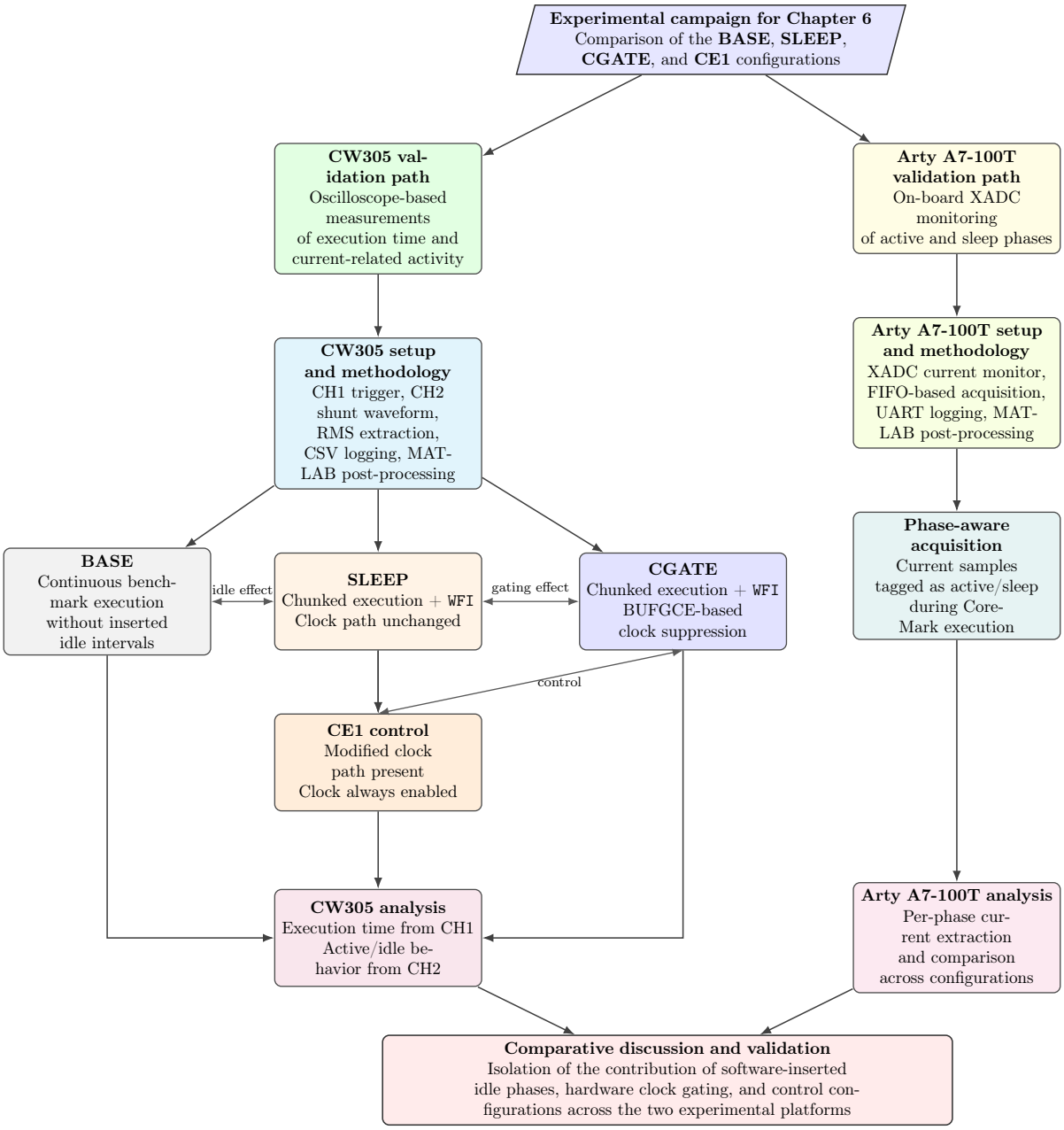


Figure 6.1: Experimental flow adopted in Chapter 6. The campaign is divided into two complementary validation paths: a CW305 oscilloscope-based analysis and an Arty A7-100T XADC-based analysis. Across the two platforms, the **BASE**, **SLEEP**, **CGATE**, and **CE1** configurations are used to separate the effects of inserted idle phases, clock-path modifications, and effective hardware clock gating.

6.1 Measurement setup on the CW305 platform

The experimental validation was carried out on a NewAE CW305 Artix-7 FPGA board, used as the hardware target for the NEORV32-based system implementing the CoreMark benchmark. The purpose of this setup was to obtain direct timing and power-related observations during benchmark execution, in order to compare the different architectural configurations discussed in this work. The physical setup and its corresponding block diagram representation are shown in Fig. 6.2.

The FPGA design was implemented in Vivado and loaded onto the CW305 board. The system clock was provided through the CW305 clock input, configured at 100 MHz, while the benchmark software was stored in the internal instruction memory of the processor. The processor data memory was also implemented internally in the FPGA fabric, so that the complete benchmark execution could be observed on-chip without relying on external memory accesses.

A dedicated trigger signal was introduced in the hardware/software co-design flow in order to delimit the benchmark execution window. At the software level, the CoreMark code was modified so that GPIO output bit 0 is driven low before the measured section, asserted high at the beginning of the benchmark execution, and driven low again at the end of the measured section. At the hardware level, this signal was connected to the external trigger output of the board by mapping `gpio_out(0)` to the top-level signal `trig_o`. In the CW305 constraint file, `trig_o` was assigned to pin `T14`, which is the board trigger pin used during the oscilloscope measurements, as illustrated in Fig. 6.2b.

The reset and communication signals were also adapted to the CW305 platform. In particular, the board reset pin was connected according to the CW305 reset behavior, and UART0 was mapped to the board serial pins in order to preserve software output and debugging capability during execution. The final top-level hardware configuration used a dedicated CW305-specific constraint file to avoid ambiguity in clock, reset, trigger, and UART routing.

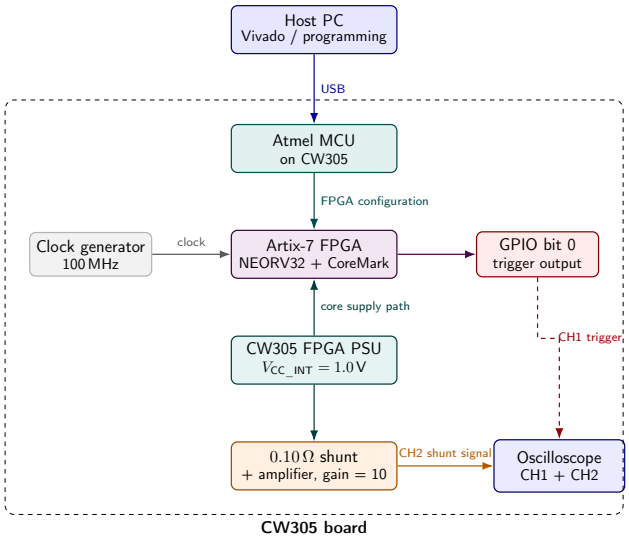
Power-related observations were obtained through the CW305 shunt-based measurement path on the FPGA core supply rail. The board version used in this work includes a 0.10 Ω shunt resistor on the VCC_INT path, together with an amplified measurement output. This measurement path makes it possible to observe activity-related variations associated with the FPGA core consumption. In the adopted setup, the FPGA core voltage was set to 1.0 V, and the measurement output was acquired with an effective voltage gain of 10 (+20 dB), as shown in Fig. 6.2a.

The oscilloscope was connected so that Channel 1 observed the external trigger signal and Channel 2 observed the shunt-related measurement output. In this way, the timing information and the power-related waveform could be correlated directly on the same acquisition. More specifically:

- **Channel 1** was used to measure the trigger pulse width, which corresponds to the execution time of the selected benchmark section;
- **Channel 2** was used to observe the activity of the FPGA core supply path during active and idle intervals.

This setup was used consistently across all the considered configurations: the baseline case, the sleep baseline case, the clock-gated implementation, and the control configuration with permanently enabled clock-gating hardware. As a result, the differences observed in the measurements can be attributed to the design modifications rather than to changes in the measurement environment.

(a) Photo of the physical CW305 measurement setup.



(b) Simplified block diagram of the measurement setup.

Figure 6.2: CW305 measurement setup. (a) Physical board with oscilloscope connections. (b) Block diagram: the host PC configures the Artix-7 FPGA via the on-board Atmel MCU over USB; the oscilloscope simultaneously acquires the GPIO-based trigger (CH1) and the amplified shunt signal from the FPGA core supply path (CH2).

6.1.1 Trigger generation and benchmark timing

In order to obtain a direct timing reference for the benchmark execution, a dedicated trigger signal was generated through the GPIO peripheral of the NEORV32 system. More specifically, GPIO output bit 0 was used as a software-controlled marker for the beginning and the end of the measured benchmark section.

The CoreMark source code was modified so that the GPIO output is forced low before the beginning of the timed execution, asserted high immediately before entering the benchmark section of interest, and driven low again immediately after the end of that section. As a consequence, the resulting waveform observed on the external trigger pin, as illustrated in Fig. 6.3, is a rectangular pulse whose duration corresponds to the execution time of the benchmark code included between the two transitions.

From a measurement point of view, this approach provides a simple and robust way to associate the oscilloscope time axis with the actual benchmark activity. The rising edge of the trigger identifies the start of the measured computation, while the falling edge identifies its completion. Therefore, the pulse width measured on Channel 1 can be directly interpreted as the benchmark execution time for the selected configuration.

This mechanism was used consistently throughout the experimental campaign. In the baseline case, the trigger pulse encloses the full benchmark execution without any inserted idle intervals. In the other configurations, the same trigger signal was preserved so that all timing results remain directly comparable, even when software-controlled sleep phases or hardware clock-gating logic are introduced.

The use of an externally observable trigger signal is particularly useful because it allows the timing information to be correlated with the power-related waveform acquired on the second oscilloscope channel. In this way, the execution interval can be identified visually and the dynamic activity of the system can be interpreted in relation to the

actual benchmark phases. Conceptually, the implemented software instrumentation can be summarized as shown in Listing 6.1:

```
GPIO(0) = 0;
start_time();
GPIO(0) = 1;
benchmark_execution();
GPIO(0) = 0;
stop_time();
```

Listing 6.1: GPIO-based trigger instrumentation used to delimit the benchmark execution window.

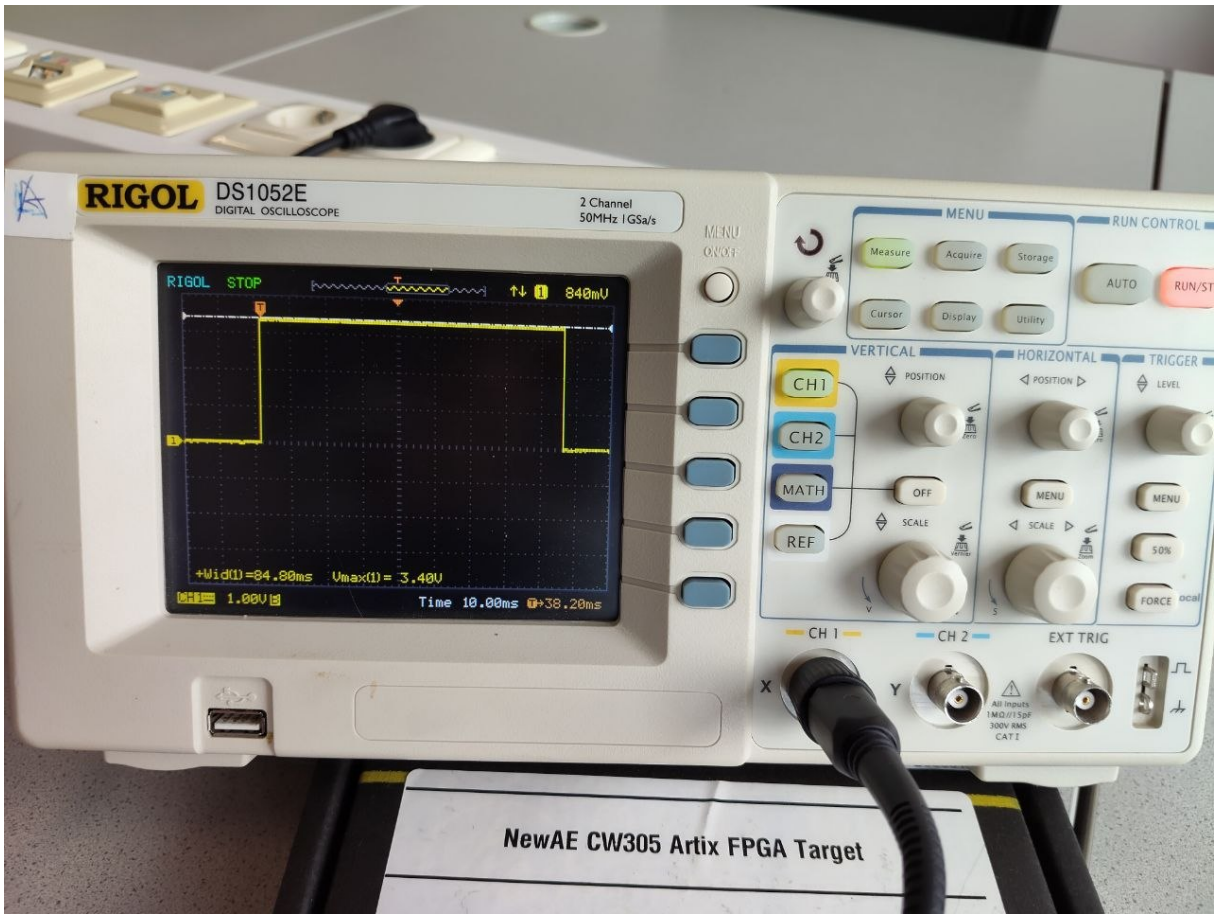


Figure 6.3: Example trigger pin behavior

6.2 Tested configurations

In order to evaluate the effect of software-controlled idle phases and hardware clock gating on the NEORV32-based implementation of CoreMark, four different configurations were considered during the experimental campaign. These configurations were not introduced independently of one another, but rather as a progressive sequence aimed at isolating the contribution of each design choice.

For clarity, the configurations are hereafter referred to using the following shorthand notation:

- **BASE**: baseline configuration without idle intervals;
- **SLEEP**: software-modified configuration with inserted idle intervals;
- **CGATE**: clock-gated configuration with active clock suppression;
- **CE1**: control configuration with `CE = '1'` (no clock gating).

The first configuration (**BASE**) represents the nominal execution case and provides the reference timing behavior of the system. The second (**SLEEP**) introduces explicit idle intervals at software level without modifying the hardware clock distribution. The third (**CGATE**) applies the actual clock-gating strategy proposed in this work. Finally, the fourth (**CE1**) preserves the modified clock-tree structure while disabling the gating action itself, thus acting as a control case.

6.2.1 Baseline configuration (BASE)

The baseline configuration corresponds to the standard execution of the CoreMark benchmark on the NEORV32 system without any intentionally inserted idle interval between benchmark operations. In this case, the processor executes the benchmark continuously, and the trigger signal simply delimits the full measured benchmark section.

This configuration provides the reference timing behavior of the system under normal operating conditions. In particular, it allows the execution time to be measured as a function of the number of benchmark iterations and therefore serves as the nominal comparison point for the subsequent configurations.

From an architectural point of view, the baseline configuration does not include software-induced sleep phases and does not rely on hardware clock gating. Therefore, it represents the conventional active execution case against which more power-oriented solutions can be compared.

6.2.2 Sleep-aware configuration (SLEEP)

The sleep baseline configuration was introduced to separate the effect of software-controlled idle phases from the effect of hardware clock gating. In this case, the benchmark execution was divided into smaller chunks, and a sleep interval was inserted between consecutive chunks by using a WFI-based mechanism supported by the machine timer interrupt.

The key point is that, in this configuration, the processor enters a sleep state from the software point of view, but the clock is still physically distributed to the CPU subsystem. In other words, the processor is not performing useful benchmark computation during the inserted idle intervals, yet the clock tree remains active.

From a circuit-level perspective, this implies that although the internal data-path activity is reduced due to the suspension of instruction execution, the clock signal continues to toggle across the sequential elements of the processor. As a consequence, a significant portion of dynamic power consumption is still present, since clock distribution and clock-driven switching in flip-flops remain active. In addition, a residual level of activity is required to support the sleep mechanism itself, including timer operation, interrupt handling, and wake-up control logic.

This configuration is essential because it provides a fair intermediate reference between the fully active baseline case and the actual clock-gated solution. It makes it possible

to determine whether a reduction in measured activity is due only to reduced software execution density or to the real suppression of clock switching in the hardware.

6.2.3 Clock-gated configuration (CGATE)

The clock-gated configuration implements the actual low-power solution investigated in this thesis. The same chunked benchmark structure and software-controlled sleep intervals used in the sleep baseline configuration were preserved, but in this case the CPU clock was additionally controlled through a BUFGCE-based gating structure.

The enable condition of the gated clock was derived from the processor sleep status together with the wake-up sources required to correctly resume execution, such as timer and interrupt events. As a result, during the idle intervals the CPU clock can be effectively disabled, while normal clock activity is restored whenever the processor must resume operation.

This configuration is the main experimental target of the work. Its purpose is to verify whether the explicit insertion of idle phases, when combined with actual hardware clock suppression, leads to a measurable reduction in the observed activity on the FPGA core supply path.

6.2.4 Control configuration (CE1)

A further control configuration was also considered in order to validate the interpretation of the experimental results. In this case, the BUFGCE primitive remained instantiated in the design, but its clock-enable input was forced permanently high, i.e., `CE = '1'`.

Under this condition, the modified clock path is preserved, but no actual clock gating takes place. Therefore, the processor clock is continuously propagated even though the design still includes the same structural elements introduced for the gated implementation.

This case is relevant because inserting a BUFGCE may slightly modify the implemented clock network even when the clock is never disabled. Possible secondary effects include changes in clock routing, clock insertion delay, clock skew, fanout distribution, and the capacitive load seen by the clock network. These effects could in principle modify the measured switching activity independently of the actual gating action.

The purpose of this configuration is not to provide a low-power solution, but to act as a control experiment. More specifically, it allows one to verify whether any observed benefit in the gated case is genuinely caused by effective clock suppression rather than by secondary effects due to the insertion of the BUFGCE element or to changes in the clock routing structure.

6.2.5 Comparative role of the four configurations

The four configurations can be interpreted as complementary steps in the validation flow. The baseline configuration provides the nominal execution reference. The sleep baseline configuration introduces idle phases without hardware clock suppression. The clock-gated implementation evaluates the actual benefit of the proposed technique. Finally, the **CE1** case verifies that the measured differences are due to real clock suppression rather than to the mere presence of the modified clock network.

For this reason, the most meaningful comparisons in the discussion of the results are not limited to baseline versus gated. The comparison between **SLEEP** and **CGATE** is

the most relevant one to isolate the effect of clock gating during idle phases, while the comparison between **CGATE** and **CE1** provides an additional consistency check on the physical interpretation of the measured data.

Table 6.1: Matrix summary of the tested configurations.

	Clock gating disabled	**Clock gating enabled**
SW sleep disabled	**BASE**: continuous benchmark execution without explicit idle intervals.	—
SW sleep enabled	**SLEEP**: `WFI`-based idle intervals without effective hardware clock suppression. **CE1**: BUFGCE path present, but `ce_cpu = '1'` and the CPU clock is always propagated.	**CGATE**: `WFI`-based idle intervals with dynamic BUFGCE-based CPU clock suppression.

6.3 Oscilloscope-Based Measurement Methodology

The experimental analysis was carried out by combining oscilloscope-based acquisitions with offline data processing. For each tested configuration, the benchmark was executed on the CW305 platform while two signals were observed simultaneously: the trigger signal associated with the measured benchmark section and the shunt-related signal associated with the FPGA core activity.

The trigger waveform, acquired on Channel 1, was used to determine the execution interval of the benchmark section under analysis. More precisely, the time difference between the rising and falling edges of the trigger pulse was used as the execution time of the measured section. This quantity was evaluated for different numbers of CoreMark iterations in order to study the timing behavior of the system under the considered configurations.

The *iterations* parameter defines how many times the CoreMark workload is executed within the measured benchmark window. Measurements were collected for multiple iteration counts in order to verify the scalability of the timing results and to evaluate whether the activity-related indicators remain consistent when the observation window becomes longer.

Low iteration counts, such as 1 or 2 iterations, provide short execution windows and are useful to verify the trigger mechanism and the visibility of individual active/idle transitions. However, these measurements are more sensitive to oscilloscope noise, trigger uncertainty, and transient effects. Higher iteration counts, such as 10, 20, and 50 iterations, provide longer observation windows and therefore more stable RMS estimates, since the measured activity is averaged over a larger number of benchmark operations and sleep intervals.

The selected iteration values were chosen as a compromise between measurement resolution and acquisition time. They cover both short executions, where transient effects are more visible, and longer executions, where the average behavior of the system becomes clearer. This also makes it possible to verify that the execution time scales approximately

linearly with the workload and that the measured activity trends are not artifacts of a single benchmark duration.

The second oscilloscope channel was used to observe the amplified signal derived from the shunt-based measurement path on the VCC_INT rail. Since this signal was often acquired in AC coupling mode, the measured values cannot be interpreted as absolute DC power values. Instead, the corresponding RMS measurements were used as comparative indicators of dynamic activity. In particular, two values were considered:

- an **active** RMS value, measured during the benchmark execution interval;
- an **idle** RMS value, measured during an idle interval, when such an interval was explicitly present in the considered configuration.

For the baseline configuration, the active RMS value was associated with the benchmark execution window delimited by the trigger signal. For the configurations including software-inserted sleep phases, the idle RMS value was measured specifically during the sleep interval between consecutive benchmark chunks, since this interval is the one of actual interest for evaluating the impact of clock gating.

To improve robustness, multiple measurements were collected for the same setup. The experimental data were organized in comma-separated value (CSV) files, one for each considered configuration. These files contain the measured timing and RMS values associated with the different iteration counts and repeated acquisitions.

The CSV files were then processed offline using dedicated MATLAB scripts developed for this work. These scripts were used to:

- import the measured data;
- compute derived quantities such as active-idle RMS differences;
- estimate equivalent current and equivalent power-related indicators from the measured shunt signal;
- generate plots showing the dependence of execution time and activity-related quantities on the number of benchmark iterations;
- export processed tables for further comparison.

The use of a scripted post-processing flow was important to ensure consistency across the different configurations. It also made it possible to compare the baseline, sleep baseline, clock-gated, and `CE = '1'` control cases using the same analysis procedure and the same derived metrics.

It should be emphasized that, due to the AC-coupled acquisition on the shunt-related signal, the resulting current- and power-related quantities are not intended as precise absolute average consumption values of the complete FPGA system. Rather, they are used as equivalent comparative indicators, suitable for highlighting the relative differences among the tested configurations under identical measurement conditions.

A more precise measurement of the average power consumption of the complete FPGA system would require a DC-coupled acquisition of the supply current over the full execution interval, including both active and idle phases. In practice, this would require measuring the voltage drop across the shunt resistor with sufficient bandwidth and resolution, preserving the DC component of the signal, and converting it into current using

the known shunt resistance and amplifier gain. The instantaneous power could then be computed as

$$P(t) = V_{\text{CC\_INT}} \cdot I(t),$$

and the average power over the benchmark interval could be obtained as

$$P_{\text{avg}} = \frac{1}{T} \int_0^T P(t)\, dt.$$

In addition, measuring the complete FPGA system would require considering all relevant supply rails, not only the VCC_INT core rail. This includes auxiliary and I/O supplies, depending on the board configuration and active interfaces. Therefore, the measurements reported in this work should be interpreted as comparative indicators of dynamic activity on the FPGA core supply path, rather than as absolute power measurements of the entire board or FPGA system.

6.4 Limitations of the Oscilloscope-Based Measurements

The experimental characterization carried out on the CW305 platform relies on an external oscilloscope to observe the current consumption of the device under test (DUT). While this approach provides direct access to the temporal evolution of the supply current, it inherently introduces a number of limitations that must be carefully considered when interpreting the results.

The measurement principle is based on monitoring the voltage variation associated with the shunt resistor placed on the VCCINT supply rail. Under ideal DC-coupled and calibrated conditions, this signal could be converted into current through Ohm's law and then into instantaneous power. In the present setup, however, the acquired waveform is used primarily as a comparative activity-related indicator.

A trigger signal generated by the DUT is used to delimit the execution window of the benchmark, allowing a separation between active and idle phases.

Although conceptually straightforward, this methodology is strongly influenced by the characteristics of the acquisition instrument. In particular, the oscilloscope used in this work belongs to a class of general-purpose digital oscilloscopes with limited vertical resolution (typically 8 bits) and finite analog bandwidth. These constraints directly impact the fidelity of the acquired waveform.

A first source of uncertainty arises from quantization effects. With an 8-bit vertical resolution, the input signal is discretized into a relatively small number of levels. When the voltage drop across the shunt resistor is small—as is the case for low-current or idle conditions—the quantization step becomes comparable to the signal amplitude. This leads to a reduced signal-to-noise ratio and limits the ability to resolve fine variations in current consumption. As a result, the measured RMS values may exhibit noticeable variability, especially for short acquisition windows.

A second critical limitation is related to the bandwidth of the oscilloscope and of the measurement chain. The switching activity of a digital system implemented on FPGA contains significant high-frequency components, associated with clock edges and internal transitions. However, the oscilloscope effectively behaves as a low-pass system, attenuating fast transients and smoothing the measured waveform. This results in an underestimation of peak currents and a distortion of the true dynamic profile of the DUT.

Additional distortion is introduced by the probe itself. The input stage of the oscilloscope, together with the probe, contributes parasitic capacitance and resistance that load the measured node. This effect becomes increasingly relevant at higher frequencies, where it further filters the signal and modifies the observed waveform. Consequently, the measured voltage across the shunt resistor does not perfectly represent the actual instantaneous current flowing through the device.

The shunt resistor, which is assumed to be ideal in the reconstruction process, also contributes to the overall uncertainty. In practice, its value is affected by manufacturing tolerance, temperature variations, and parasitic inductance. These non-idealities introduce both systematic and frequency-dependent errors, which are difficult to compensate without a dedicated calibration procedure.

Another non-negligible aspect concerns the temporal alignment between the trigger signal and the actual computational activity. The trigger is generated at the software level and propagates through the FPGA fabric before being observed externally. This introduces a delay that may slightly shift the apparent boundaries of the active region. While this effect is relatively small compared to the overall execution time, it contributes to the uncertainty in separating active and idle intervals, especially for short benchmarks.

Finally, the measurement is subject to noise and environmental disturbances, including power supply fluctuations and electromagnetic interference. These factors introduce variability in the acquired waveforms and can affect the repeatability of the measurements, particularly when the signal amplitude is low.

Taken together, these limitations imply that the oscilloscope-based approach does not provide an accurate estimation of the absolute power consumption of the system. In particular, peak values are generally underestimated and fast transient effects are partially lost. However, the methodology remains highly effective for comparative analysis. Since all configurations are measured under identical conditions—same hardware platform, routing, supply, and acquisition setup—the systematic errors affect all measurements in a consistent way.

For this reason, the results presented in this work should be interpreted primarily in a relative sense. Differences between configurations, such as the reduction of idle activity obtained through clock gating, can be considered reliable even in the presence of measurement inaccuracies. This makes the oscilloscope-based approach a suitable tool for validating the effectiveness of architectural optimizations, while more accurate absolute measurements are deferred to subsequent analysis using on-chip monitoring techniques.

6.5 Measurement setup on the Arty A7-100T platform

A second experimental campaign was carried out on the Digilent Arty A7-100T platform in order to obtain a more integrated and repeatable measurement flow with respect to the oscilloscope-based setup used on the CW305 board. In this phase, the NEORV32-based system was again implemented on an Artix-7 FPGA and used to execute the CoreMark benchmark, but the power-related quantities were acquired through the on-board current sensing path and the FPGA XADC module.

The purpose of this second setup was not only to repeat the comparison between the considered configurations, but also to reduce the dependence on external instrumentation. In the CW305 measurements, the activity-related signal was observed through an oscilloscope and therefore affected by probe placement, vertical resolution, coupling mode, triggering conditions, and acquisition-window selection. On the Arty A7-100T

platform, instead, the current-related signal is sampled directly by the FPGA internal analog-to-digital converter and transferred to the software through the GPIO interface. The acquired samples are then printed through UART and post-processed in MATLAB.

The hardware design was implemented in Vivado and loaded on the Arty A7-100T board. The system clock was provided by the on-board 100 MHz clock source, while the benchmark application was stored in the internal instruction memory of the NEORV32 processor. The internal data memory was also enabled, so that the benchmark execution did not rely on external memory transactions.

As in the previous experimental phase, a dedicated trigger signal was used to identify the benchmark execution window. At software level, GPIO output bit 0 was driven high during the measured computation phase and low outside it. At hardware level, this signal was connected to the top-level output `trig_o`, allowing the same trigger-based timing strategy to be preserved. In the Arty A7 constraint file, this signal was mapped to a PMOD pin, so that it could still be observed externally if needed.

The main difference with respect to the CW305 setup concerns the power measurement path. The Arty A7-100T board provides a current-sense signal associated with the FPGA core supply rail. This signal, referred to as `ISNS0V95`, is connected to an auxiliary analog input of the XADC. The corresponding differential analog pins, `isns0v95_p` and `isns0v95_n`, were exposed at the top level and routed to the XADC primitive through a dedicated VHDL module.

The XADC module continuously converts the selected auxiliary channel and provides a 12-bit digital output. This value is then connected to the NEORV32 GPIO input port, together with a validity flag indicating the availability of a new sample. More specifically, GPIO input bits `gpio_i(11 downto 0)` carry the raw XADC value, while `gpio_i(12)` carries the sample-valid signal. The benchmark software reads these values during the selected measurement phases and sends the resulting data to the host PC through UART.

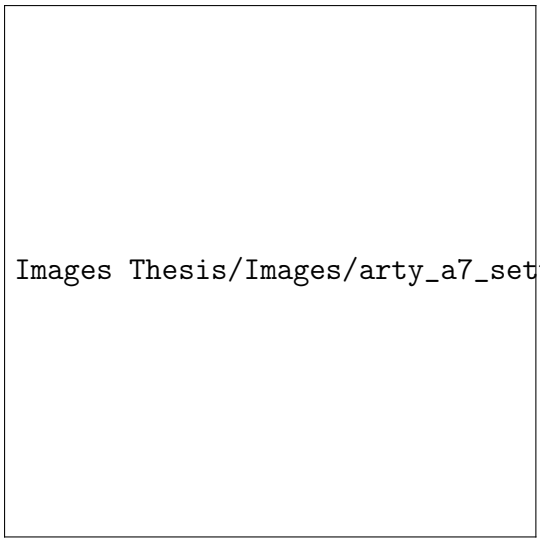
The UART log therefore represents the raw acquisition file used for MATLAB post-processing. Each valid measurement sample is stored as a CSV-like row containing the phase label, the sample index, the raw XADC value, and the reconstructed current value. This allows the measured data to be grouped according to the execution phase, for example idle-before, active, idle-after, or, in the clock-gated case, active chunks and sleep intervals. In the final implementation, the phase information is not assigned only during software readout. Instead, the hardware acquisition logic stores each XADC sample together with the phase code active at the acquisition time. Each FIFO entry therefore contains both the sampled phase code and the raw XADC value, i.e.,

$$\{\text{phase}[1:0], \text{raw}[11:0]\}.$$

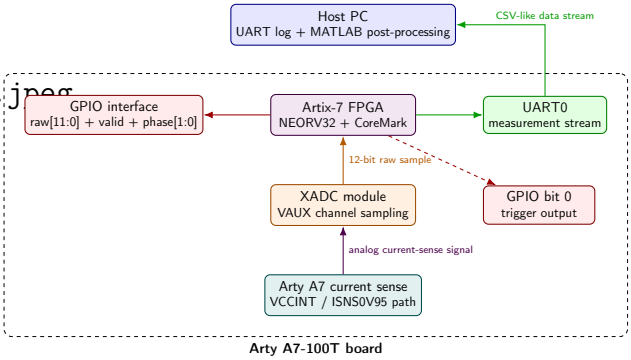
This avoids assigning an incorrect phase to samples that were already present in the FIFO before the UART dumping operation.

At the GPIO interface, the raw XADC sample is exposed on `gpio_i(11 downto 0)`, the FIFO/sample-valid flag on `gpio_i(12)`, and the sampled phase code on `gpio_i(14 downto 13)`. On the output side, the processor controls the FIFO pop signal, the phase code driven to the acquisition logic, and the capture-enable signal through dedicated GPIO outputs.

The adopted setup is shown in Fig. 6.4. Compared with the previous oscilloscope-based acquisition, this configuration provides a more automated measurement flow: the FPGA executes the benchmark, samples its own current-monitoring signal through the XADC, and streams the resulting measurements to the host PC for analysis.



(a) Photo of the Arty A7-100T measurement setup.



(b) Simplified block diagram of the Arty A7-100T measurement setup.

Figure 6.4: Arty A7-100T measurement setup. The FPGA samples the on-board current-sense signal through the XADC, exposes the converted value to the NEORV32 processor through GPIO, and sends the collected samples to the host PC through UART for MATLAB post-processing.

6.6 Tested configurations on the Arty A7-100T platform

The same set of processor configurations considered in the CW305 experimental campaign was also evaluated on the Arty A7-100T platform. The purpose of this second phase was not to redefine the validation strategy, but to verify the same architectural behavior using a different measurement flow based on the on-board current-sensing path and XADC-based acquisition.

For this reason, the four configurations are kept unchanged: the continuous baseline execution, the sleep-aware execution without effective clock gating, the actual clock-gated implementation, and the `CE = '1'` control case. The baseline configuration provides the reference behavior of the processor during uninterrupted CoreMark execution. The sleep-aware configuration introduces software-defined idle intervals between benchmark chunks, while keeping the processor clock physically active. The clock-gated configuration combines the same idle intervals with hardware clock suppression through the BUFGCE-based mechanism. Finally, the `CE = '1'` configuration preserves the modified clock path but forces the clock enable permanently high, thus allowing the effect of actual clock suppression to be separated from the structural effect of inserting the BUFGCE primitive.

In the Arty A7-100T measurements, these configurations are therefore used with the same comparative role already defined for the CW305 analysis. The most relevant comparison remains the one between the sleep-aware and the clock-gated cases, because it isolates the contribution of hardware clock suppression during software-defined idle phases. The `CE = '1'` case is again used as a control reference, while the continuous baseline provides the nominal execution point without inserted idle intervals.

The main difference to the previous setup lies in the acquisition method. Instead of

manually extracting RMS values from oscilloscope windows, the current-related samples are collected directly through the XADC and exported through UART. This makes it possible to associate the measured samples with explicit phase labels, such as active chunks and idle intervals, and to process the resulting data in a more systematic way in MATLAB.

6.7 XADC Limitations and Measurement Accuracy

Although the use of the on-chip XADC improves the repeatability and automation of the acquisition flow with respect to the oscilloscope-based setup, the obtained measurements are still affected by several limitations related to the FPGA analog front-end, the current sensing path, and the adopted sampling methodology.

The first limitation is related to the finite resolution of the XADC converter. The measurements are based on a 12-bit quantization over a full-scale voltage range of approximately 1 V. Consequently, the minimum detectable voltage variation is

$$\Delta V_{\mathrm{LSB}} = \frac{V_{\mathrm{FS}}}{2^{12}} = \frac{1\ \mathrm{V}}{4096} \approx 244\ \mu\mathrm{V}.$$

Given the gain and shunt characteristics of the Arty A7 sensing path, this voltage quantization step translates into a finite current resolution. The monitored signal can be expressed as

$$V_{\mathrm{ADC}} = G \cdot R_{\mathrm{shunt}} \cdot I_{\mathrm{core}},$$

with

$$G = 50, \qquad R_{\mathrm{shunt}} = 10\ \mathrm{m}\Omega.$$

Therefore, the reconstructed core current is

$$I_{\mathrm{core}} = \frac{V_{\mathrm{ADC}}}{G \cdot R_{\mathrm{shunt}}}.$$

With the conversion model adopted in the firmware and MATLAB post-processing, the current is reconstructed directly from the raw XADC value as

$$I_{\mathrm{core}} = \frac{2 \cdot \mathrm{raw}}{4096}.$$

This corresponds to assuming a 0–2 A full-scale range for the monitored current path. Under this model, the current resolution is approximately

$$\Delta I_{\mathrm{LSB}} = \frac{2\ \mathrm{A}}{4096} \approx 488\ \mu\mathrm{A}.$$

As a consequence, current variations close to this threshold may become difficult to distinguish, especially during low-activity idle phases where the signal amplitude is relatively small.

The instantaneous power associated with the monitored FPGA core rail is reconstructed as

$$P(t) = V_{\mathrm{CCINT}} \cdot I_{\mathrm{core}}(t),$$

where, for the considered platform,

$$V_{\mathrm{CCINT}} \approx 1 \text{ V}.$$

However, because of the finite ADC resolution and the limited sampling granularity, the reconstructed power values should be interpreted as sampled activity-related estimates, not as exact continuous-time power waveforms.

A second important limitation concerns the effective sampling rate of the acquisition process. Although the XADC itself can operate at relatively high conversion frequencies, the effective sampling rate observed in the collected data is constrained by the interaction between the XADC peripheral, the GPIO interface, the FIFO readout mechanism, the UART transmission bandwidth, and the execution overhead of the benchmark application. Therefore, the collected samples do not represent a continuous-time reconstruction of the instantaneous current waveform, but rather a discrete-time observation of the activity evolution.

This limitation is particularly relevant when analyzing fast transient phenomena. FPGA switching activity may include short-duration current spikes associated with clock edges, combinational transitions, memory accesses, and internal routing activity. Such events can occur on time scales much shorter than the effective acquisition interval and may therefore not be fully captured. For this reason, the measured current values should be interpreted as sampled indicators of the activity level rather than exact instantaneous current waveforms.

Another source of uncertainty is associated with the analog front-end of the board itself. The current-sense amplifier and the shunt resistor introduce finite tolerances, offset errors, temperature dependence, and non-ideal linearity. These effects influence the accuracy of the reconstructed current values and may introduce systematic deviations between the measured current and the actual current consumption.

In addition, the XADC measurements are limited to the monitored VCCINT rail. Therefore, the obtained power-related quantities do not represent the total power consumption of the FPGA board, but only the activity associated with the FPGA core supply path. Dynamic contributions originating from auxiliary rails, I/O banks, clock management circuitry, or external interfaces are not directly included in the reported measurements.

An additional limitation arises from the acquisition strategy itself. Since the benchmark software is responsible for controlling the experiment and transmitting the collected samples, the measurement process introduces a certain overhead. Although this overhead is small compared with the overall benchmark execution time, it still perturbs the system activity to some extent. For this reason, the measurements should not be interpreted as completely non-intrusive observations of the system behavior.

Despite these limitations, the XADC-based methodology remains suitable for comparative analysis. All the considered configurations were evaluated under identical hardware conditions, identical supply settings, identical acquisition procedures, and identical post-processing flows. Therefore, most systematic errors remain approximately constant across the different experimental cases. As a consequence, the measured differences between the **BASE**, **SLEEP**, **CGATE**, and **CE1** configurations can still be considered meaningful indicators of relative variations in switching activity and dynamic power behavior.

For this reason, the measurements reported in the following sections should primarily be interpreted as comparative activity measurements rather than as absolute high-precision power estimations of the complete FPGA system.

Chapter 7

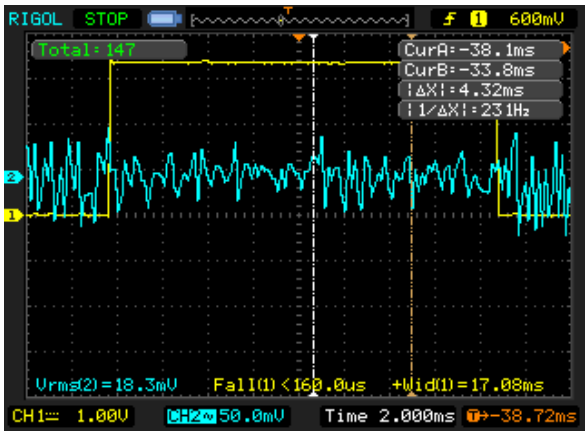
Experimental Results and Discussion

7.1 CW305 Oscilloscope-Based Results

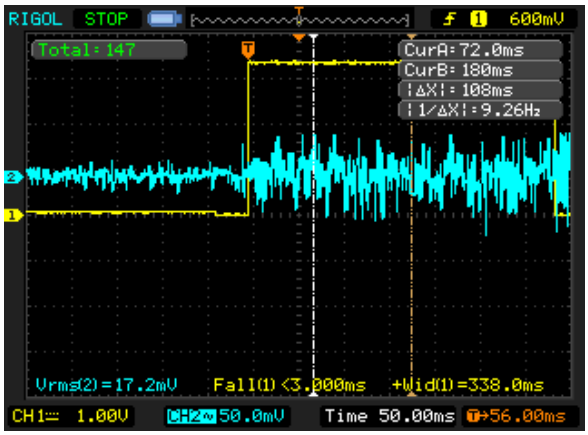
The first part of the experimental analysis was performed on the CW305 platform using oscilloscope-based measurements. In this setup, the GPIO trigger signal was used to identify the benchmark execution window, while the shunt-related waveform was used as a comparative indicator of activity on the FPGA core supply path. The following subsections present the BASE, SLEEP, CGATE, and CE1 configurations measured with this methodology.

7.1.1 Baseline configuration (BASE)

The baseline configuration represents the reference case in which the CoreMark benchmark is executed without any software-defined idle intervals and without any clock-gating mechanism. In this condition, the processor operates continuously, and the clock signal is always propagated through the system.



(a) Single-iteration acquisition.



(b) Twenty-iteration acquisition.

Figure 7.1: Oscilloscope screenshots for the baseline configuration.

The oscilloscope figure Fig. 7.1 provide a qualitative view of the system behavior. The trigger signal defines the execution interval, while the shunt-related waveform reflects the activity of the FPGA core supply.

In this configuration, the activity is continuous over the entire execution window, and no structured separation between active and idle phases is present. As a consequence, the

measured signal does not exhibit distinct low-activity intervals, since the clock is always active and continuously drives switching activity in the system.

These observations establish the baseline case as a reference condition for evaluating the impact of software-defined idle phases and hardware clock-gating techniques.

7.1.2 Sleep-aware baseline results (SLEEP)

In this configuration, the CoreMark benchmark execution was intentionally modified to include software-defined idle intervals between consecutive computation chunks. However, no hardware clock-gating mechanism was applied, and the CPU clock remained continuously active throughout both active and idle phases.

This configuration plays a crucial role in the experimental flow, as it allows isolating the effect of software-controlled idle intervals without introducing any modification to the clock distribution network. As a consequence, any observed difference between active and idle conditions cannot be attributed to clock suppression, but only to the behavioral effect of the software-induced sleep phases.

The execution time was first analyzed as a function of the number of benchmark iterations. As expected, the measured execution time increases approximately linearly with the iteration count, as shown in Fig. 7.2. However, in this case the total execution time includes not only the useful computation time, but also the inserted sleep intervals between computation chunks. This explains the larger absolute values compared to a continuous execution scenario, while preserving a consistent scaling trend.

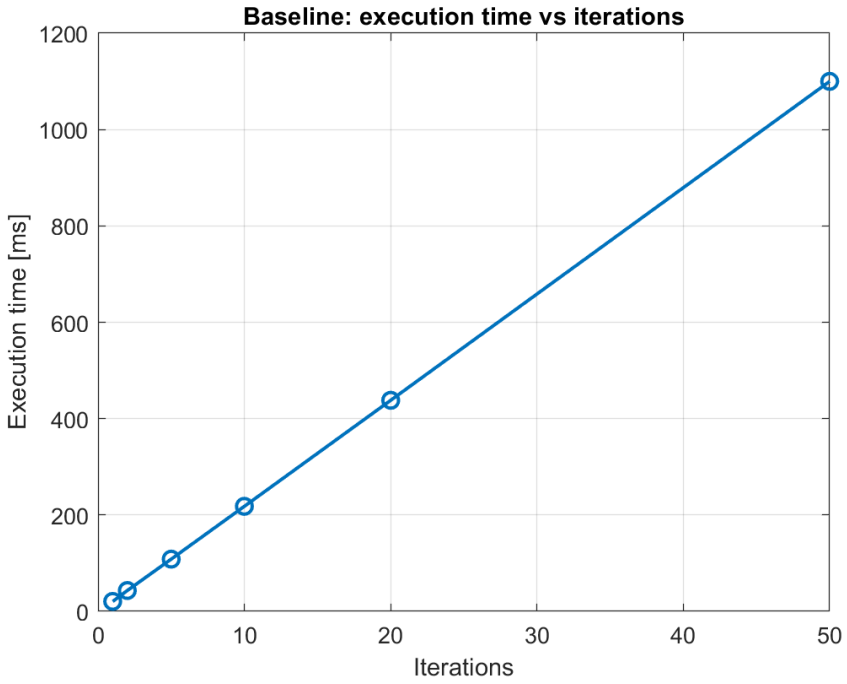
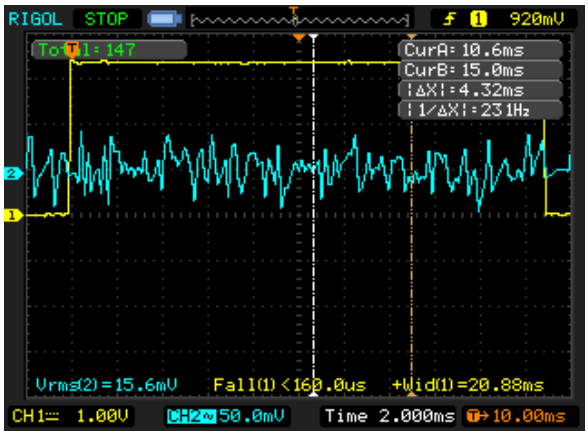


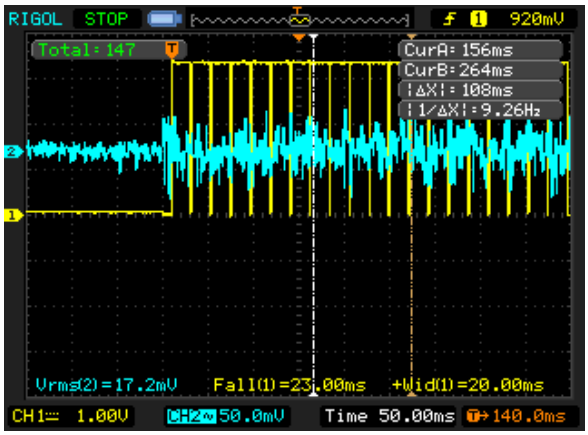
Figure 7.2: Execution time as a function of the number of CoreMark iterations for the sleep-aware baseline configuration. The measured duration includes both computation and software-defined idle intervals.

The oscilloscope acquisitions shown in Fig. 7.3 provide a qualitative confirmation of the measurement approach. The trigger signal (Channel 1) clearly identifies the execution

window, while the shunt-related waveform (Channel 2) reflects the activity on the FPGA core supply path.



(a) Single-iteration acquisition.



(b) Twenty-iteration acquisition.

Figure 7.3: Oscilloscope screenshots for the sleep-aware baseline configuration. The repeated active-idle structure is introduced by software, while the clock remains continuously active.

Unlike the pure baseline case, the presence of software-defined idle intervals makes it possible to meaningfully distinguish between active and idle phases in the RMS analysis. The active RMS value is measured during the computation window, while the idle RMS value is measured during the inserted sleep interval.

As shown in Fig. 7.4, the active RMS value increases with the number of iterations, reflecting the higher switching activity associated with longer execution windows. The idle RMS value, however, does not exhibit a significant reduction, since the clock signal remains active even during the sleep phases. As a result, the difference between active and idle conditions is limited.

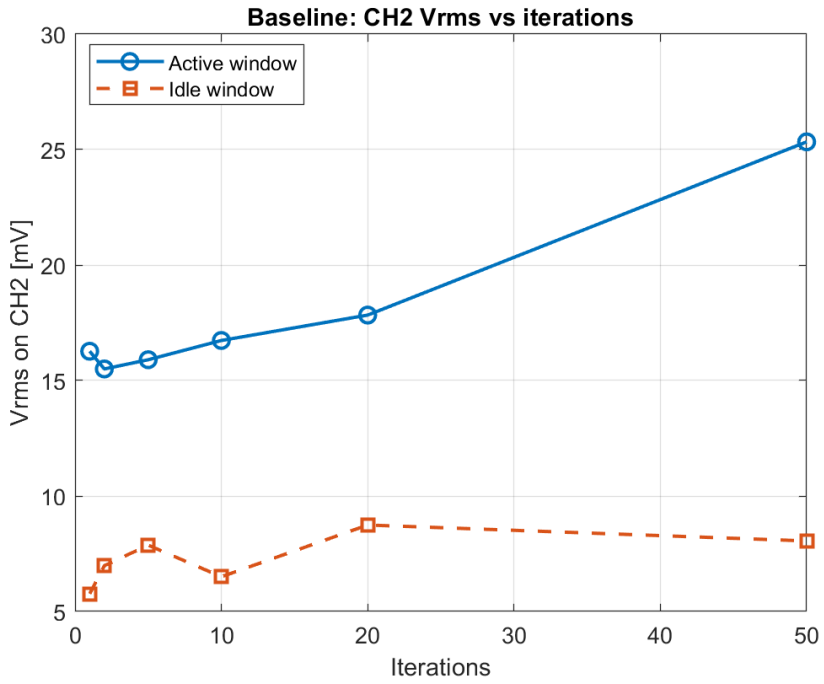


Figure 7.4: Measured RMS values for the sleep-aware baseline configuration. The idle value corresponds to a software-defined sleep interval, but the clock remains active.

The equivalent dynamic power derived from the RMS measurements is reported in Fig. 7.5. The active component follows the expected increasing trend with the workload, while the idle component remains relatively high compared to the clock-gated case.

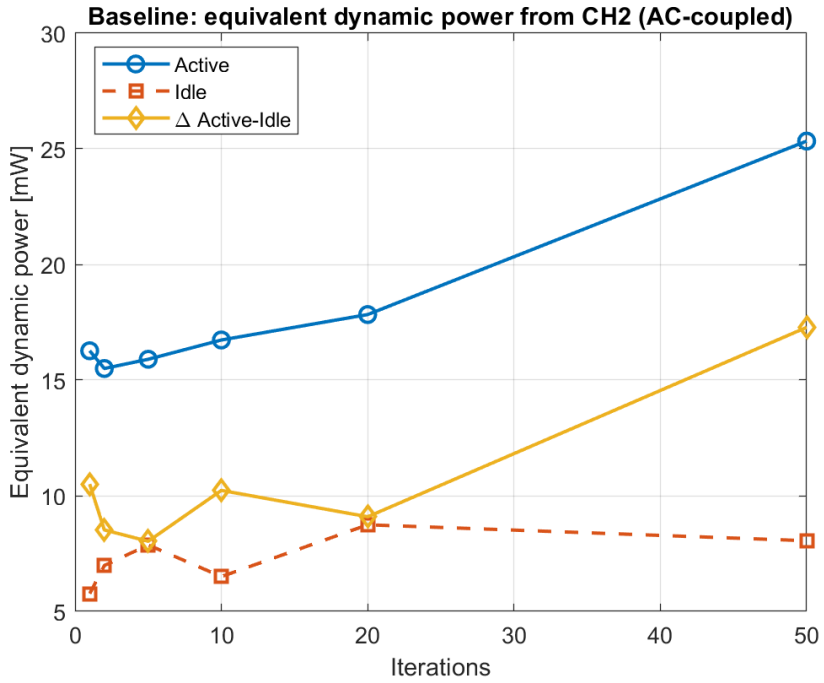


Figure 7.5: Equivalent dynamic power derived from RMS measurements for the sleep-aware baseline configuration.

This behavior highlights a key limitation of the sleep-aware baseline configuration:

although the processor enters a software-defined idle state, the absence of hardware clock gating prevents a substantial reduction of switching activity during the idle phases. In other words, the system is logically idle, but not electrically quiet.

Therefore, this configuration demonstrates that the introduction of idle intervals alone is not sufficient to achieve significant dynamic power reduction. It provides a necessary intermediate reference, which allows the effect of actual clock gating to be clearly isolated in the following sections.

Table 7.1: Experimental results for the sleep-aware baseline configuration.

Iters	T_{exec} **(ms)**	$V_{rms,A}$ **(mV)**	$V_{rms,I}$ **(mV)**	$P_{eq,A}$ **(mW)**	$T/Iter$ **(ms)**
1	20.6	16.27	5.76	16.27	20.6
2	43.2	15.5	6.98	15.5	21.6
5	108.4	15.9	7.87	15.9	21.68
10	218	16.73	6.5	16.73	21.8
20	438	17.83	8.74	17.83	21.9
50	1,100	25.33	8.05	25.33	22

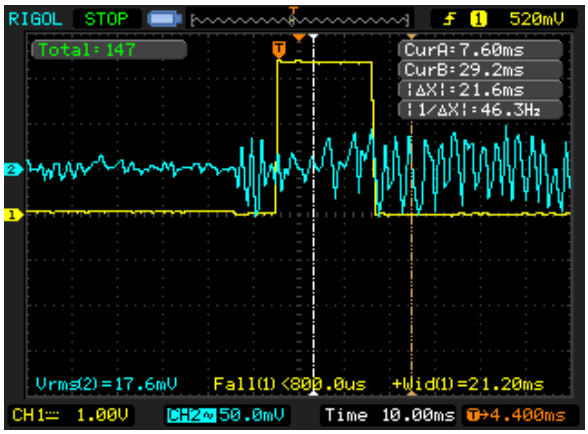
7.1.3 Clock-gated results (CGATE)

The clock-gated configuration represents the main experimental target of this work. In this case, the benchmark execution is intentionally divided into computation chunks separated by software-defined idle intervals. During these idle phases, the processor enters a sleep state and the CPU clock is effectively controlled through the BUFGCE-based gating mechanism introduced in the hardware design.

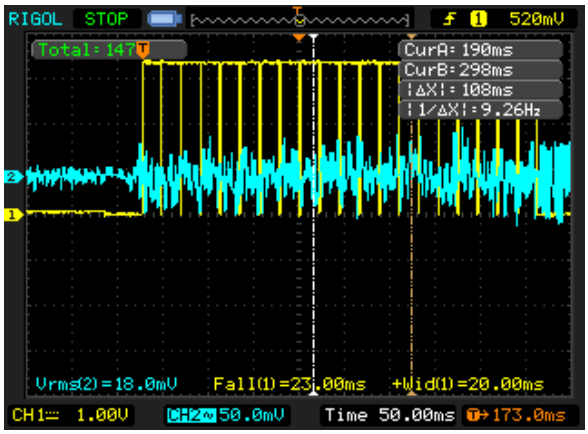
From the timing point of view, this configuration is directly comparable to the sleep-aware baseline, since both cases include the same software-defined idle intervals. The key difference is that, in the present configuration, the CPU clock is actually suppressed during the sleep phases. Therefore, any difference observed in the activity-related measurements can be attributed to hardware clock gating rather than to the software structure alone.

Unlike the baseline case, where the benchmark is executed continuously, the total execution time of the clock-gated configuration includes both the useful computation time and the inserted sleep intervals. For this reason, the measured trigger pulse width is expected to be larger than in the baseline configuration for the same number of iterations. This increase is not due to a degradation of the benchmark itself, but rather to the deliberate introduction of idle windows that are necessary to evaluate the effect of clock gating in realistic sleep conditions.

The first quantity analyzed in this configuration was again the execution time as a function of the number of benchmark iterations. Also in this case, the measured execution time increases with the iteration count, but the slope is affected by the presence of the inserted sleep intervals. The corresponding values are reported in Table 7.2 and visualized in Fig. 7.7. The resulting trend remains coherent with the benchmark structure and confirms that the trigger signal still provides a reliable timing reference even in the presence of repeated active-idle transitions.



(a) Single-iteration acquisition.



(b) Twenty-iteration acquisition.

Figure 7.6: Oscilloscope screenshots for the clock-gated configuration with different numbers of CoreMark iterations. Channel 1 (yellow) is the trigger signal generated through the GPIO-based instrumentation, while Channel 2 (cyan) is the amplified shunt-related signal used as a comparative indicator of activity on the FPGA core supply path. In the single-iteration case, the trigger window is short and the alternation between active and idle behavior is visible over a limited interval. In the twenty-iteration case, the repeated active/sleep pattern becomes much more evident, confirming that the benchmark execution is intentionally split into multiple computation chunks separated by software-defined idle intervals.

Figure 7.6 shows two representative oscilloscope acquisitions for the clock-gated configuration, corresponding to one and twenty benchmark iterations. As in the previous measurements, Channel 1 provides the external timing reference generated by the GPIO trigger, whereas Channel 2 shows the amplified shunt-related waveform associated with activity on the FPGA core supply path.

These screenshots are particularly useful because they visually highlight the behavioral difference between the continuous baseline execution and the clock-gated case. Here, the trigger signal does not simply identify one uninterrupted execution interval, but encloses a sequence of alternating active and idle phases introduced by the modified benchmark structure. This is especially clear in the twenty-iteration acquisition, where the repeated pattern confirms that the software periodically enters sleep and resumes execution under timer control.

At the same time, the Channel 2 waveform shows a corresponding modulation of activity, which is consistent with the intended alternation between useful computation and idle intervals. These acquisitions therefore provide a direct qualitative confirmation that the proposed measurement methodology is correctly capturing the behavior required for the evaluation of clock gating.

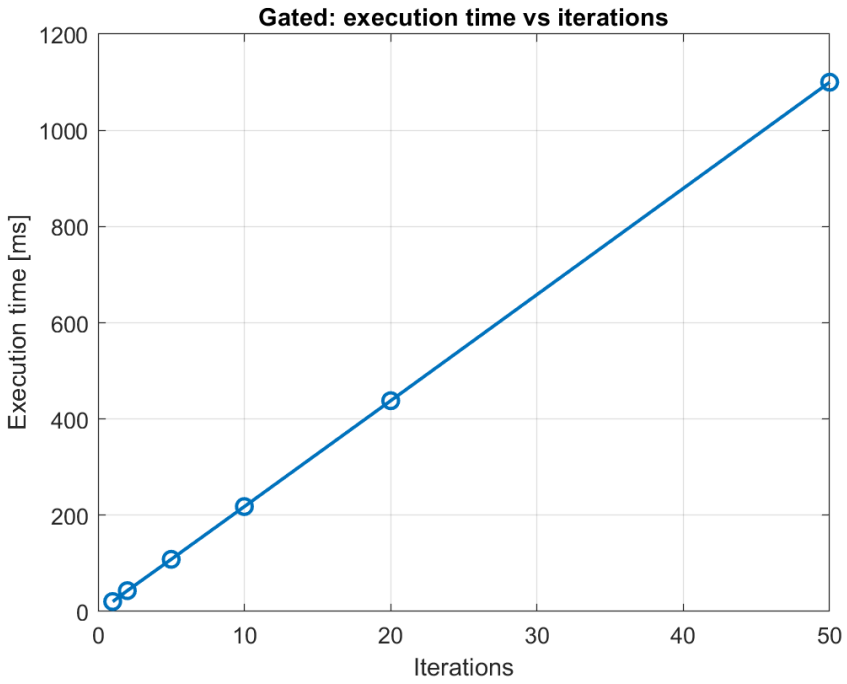


Figure 7.7: Execution time as a function of the number of CoreMark iterations for the clock-gated configuration. In this case, the total measured duration includes both benchmark execution and the software-inserted sleep intervals between computation chunks.

In the clock-gated configuration, the RMS measurements on the shunt-related signal acquire a direct physical meaning, since the distinction between active and idle intervals corresponds to actual operating states of the system.

The active RMS value is evaluated during the computation phases, while the idle RMS value is measured during the software-defined sleep intervals, in which the CPU clock is effectively gated. This allows defining a differential metric $\Delta V_{rms} = V_{rms,A} - V_{rms,I}$ that provides a comparative indicator of the activity contrast between computation and sleep phases enabled by the clock-gating mechanism.

As shown in Fig. 7.8, the idle RMS level is generally lower than the active one, indicating a reduction of switching activity during the sleep phases. This behavior is consistent with the intended effect of hardware clock gating, although some variability remains visible across the tested operating points.

The increasing trend of the active RMS value with the number of iterations is coherent with the longer observation windows, while the idle level remains comparatively lower, highlighting the effectiveness of clock suppression during non-computational intervals.

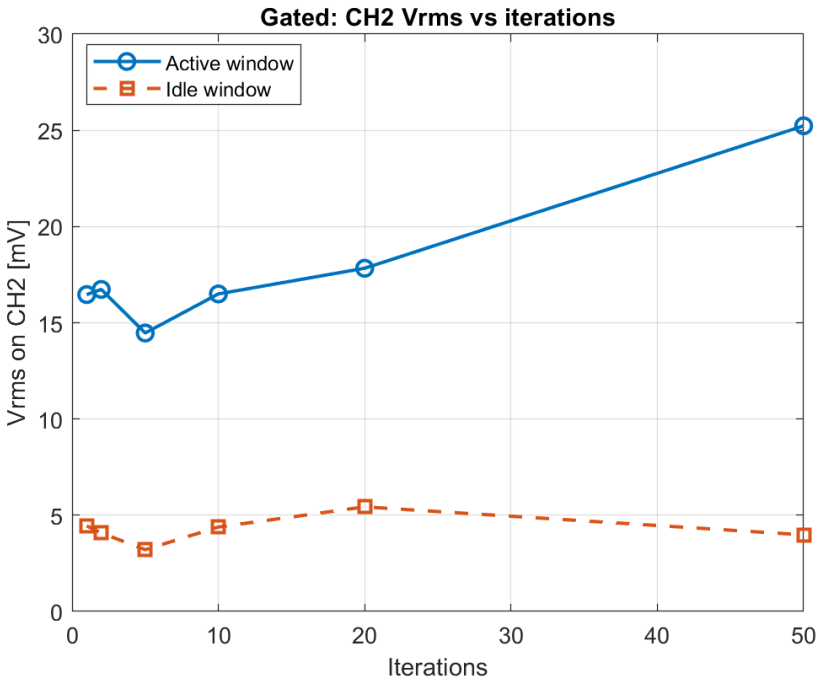


Figure 7.8: Measured active and idle RMS values for the clock-gated configuration. Unlike the baseline case, the idle value is now measured during a true software-defined sleep interval and is therefore physically meaningful for the evaluation of clock gating.

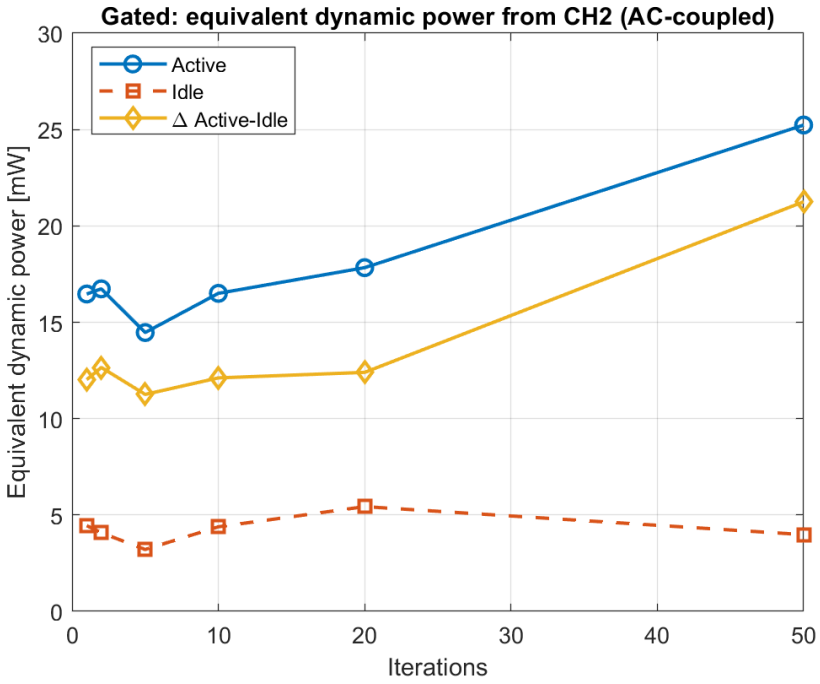


Figure 7.9: Equivalent dynamic power derived from RMS measurements for the clock-gated configuration. The difference between active and idle values highlights the reduction of dynamic activity during sleep phases.

Figure 7.9 reports the equivalent dynamic power derived from the RMS measurements. The active component follows the expected trend with increasing workload, while the idle

component remains lower and comparatively stable across the tested iteration range, reflecting the reduced switching activity during clock-gated intervals.

It is worth noting that the largest increase occurs in the active component at 50 iterations, while the idle equivalent power remains close to the lower range of the measurements. This behavior suggests that the clock-gated idle state is not strongly affected by the total benchmark duration, whereas the active window reflects the increased computational workload.

The difference ΔP_{eq} provides a compact indicator of the reduction in dynamic activity observed between active and idle phases. Overall, the results support the interpretation that hardware clock gating contributes to a measurable reduction of switching activity during software-defined sleep intervals.

As in the baseline case, the RMS quantities obtained from Channel 2 should not be interpreted as absolute average power values, since the acquisition was often performed in AC coupling mode. Nevertheless, they remain useful comparative indicators of dynamic activity. In this configuration, the most meaningful quantity is not only the active RMS level itself, but also the difference between active and idle conditions, since this difference reflects the extent to which activity is reduced during the inserted sleep phases.

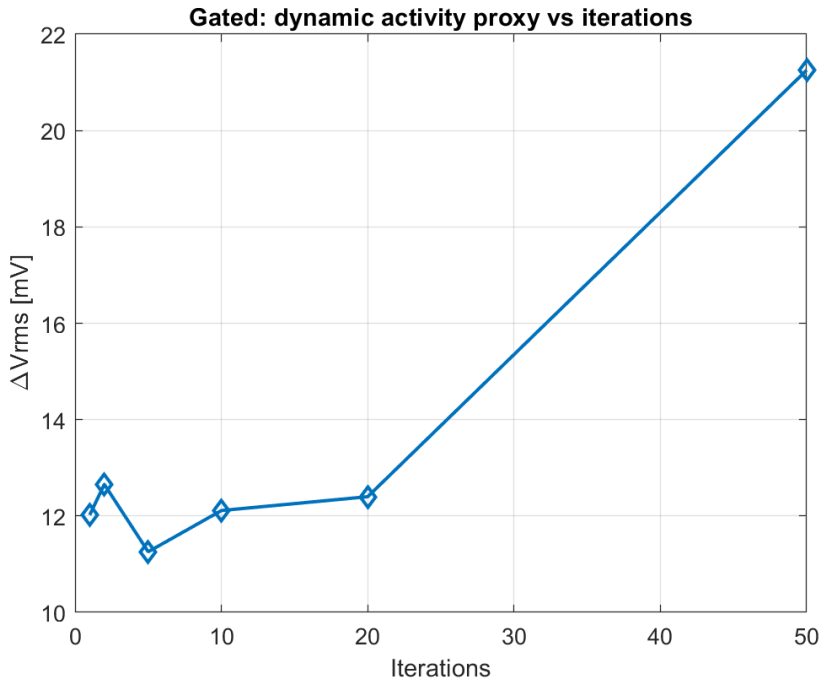


Figure 7.10: Difference between active and idle RMS values for the clock-gated configuration. This quantity provides a compact representation of the activity reduction observed between computation and sleep phases.

A further practical point concerns measurement robustness. In order to reduce the impact of waveform variability and acquisition uncertainty, repeated measurements were carried out for each tested point. The average values obtained from three repeated acquisitions are summarized in Table 7.2. This additional dataset supports the consistency of the processed results and provides a more reliable basis for the interpretation of the clock-gated behavior.

Table 7.2: Experimental results for the proposed clock-gated configuration.

Iters	T_{exec} **(ms)**	$V_{rms,A}$ **(mV)**	$V_{rms,I}$ **(mV)**	$P_{eq,I}$ **(mW)**	**Energy (μJ)**
1	20.6	16.47	4.45	4.45	247.61
2	43.2	16.73	4.08	4.08	546.77
5	108.4	14.47	3.21	3.21	1,220.22
10	218	16.5	4.39	4.39	2,640.71
20	438	17.83	5.43	5.43	5,431.2
50	1,100	25.23	3.98	3.98	23,378.67

Overall, the clock-gated configuration represents the key experimental result of this work. The measurements show that the gated configuration produces a clear separation between active and idle activity. The actual benefit of clock gating must then be confirmed through the comparison with the CE=1 control configuration.

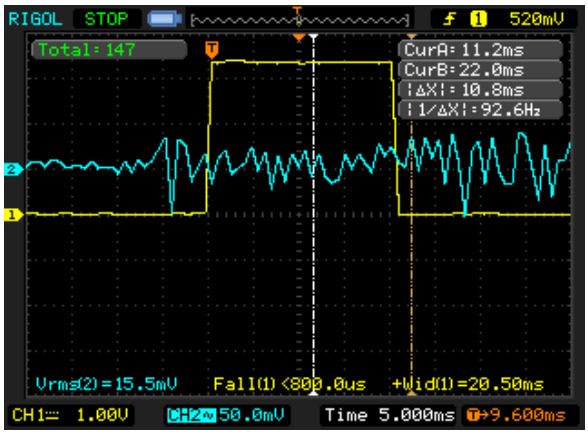
The distinction between active and idle RMS levels, together with the corresponding differential metrics, confirms that the proposed approach effectively modulates the dynamic behavior of the system without compromising correct benchmark execution. This establishes clock gating as a viable strategy for reducing unnecessary activity in FPGA-based processor implementations.

7.1.4 Control results with permanently enabled clock path (CE1)

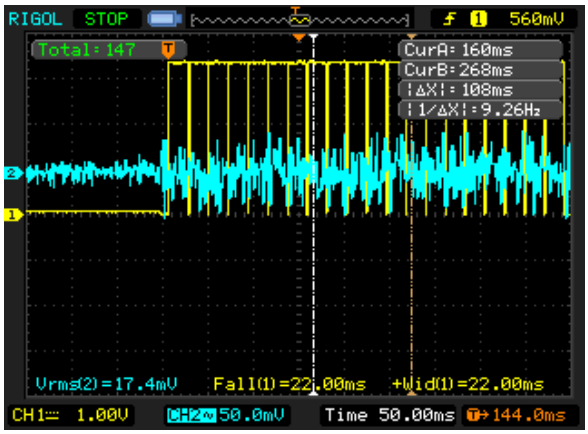
The control configuration with `CE = '1'` was introduced in order to better interpret the results obtained from the clock-gated implementation. In this case, the BUFGCE element remained instantiated in the hardware design, but its enable input was forced permanently high. As a consequence, the modified clock path is preserved, but no real clock gating takes place, since the CPU clock is continuously propagated.

The reason for considering this case is critical for the interpretation of the results. Once the BUFGCE is inserted into the clock tree, the design is no longer identical to the original baseline implementation, even if the clock is never actually disabled. Therefore, if differences are later observed between the clock-gated version and the original baseline case, it is useful to determine whether such differences are truly caused by clock suppression or whether they might be partially influenced by the mere presence of the modified clock structure.

From the timing point of view, the `CE = '1'` configuration behaves similarly to a sleep-aware execution in which the software structure includes active and idle intervals, but the hardware clock remains always available. The trigger pulse width still reflects the full measured benchmark section, and the execution time remains influenced by the inserted software sleep intervals. The linear trend confirms that the introduction of the BUFGCE element does not affect the functional execution of the benchmark, and that any difference observed in the following analysis cannot be attributed to timing artifacts.



(a) Single-iteration acquisition.



(b) Twenty-iteration acquisition.

Figure 7.11: Oscilloscope screenshots for the control configuration with `CE = '1'` and different numbers of CoreMark iterations. Channel 1 (yellow) is the trigger signal generated through the GPIO-based instrumentation, while Channel 2 (cyan) is the amplified shunt-related signal used as a comparative indicator of activity on the FPGA core supply path. Although the benchmark execution still includes software-defined idle intervals, the clock remains continuously enabled in this configuration.

Figure 7.11 reports representative oscilloscope acquisitions for the control configuration in which the clock-gating structure is present but the enable signal is permanently asserted (`CE = '1'`).

These measurements are essential for the correct interpretation of the clock-gated results. Even though the benchmark is executed using the same chunked structure with software-defined idle intervals, the CPU clock is never actually disabled in this configuration.

As a consequence, the activity observed during idle phases remains significantly above the levels measured in the clock-gated configuration, indicating that the presence of software-defined idle intervals alone does not effectively suppress switching activity. This confirms that any reduction observed in the gated configuration is effectively due to the suppression of clock switching, rather than to the mere presence of idle intervals or to structural modifications in the clock distribution network.

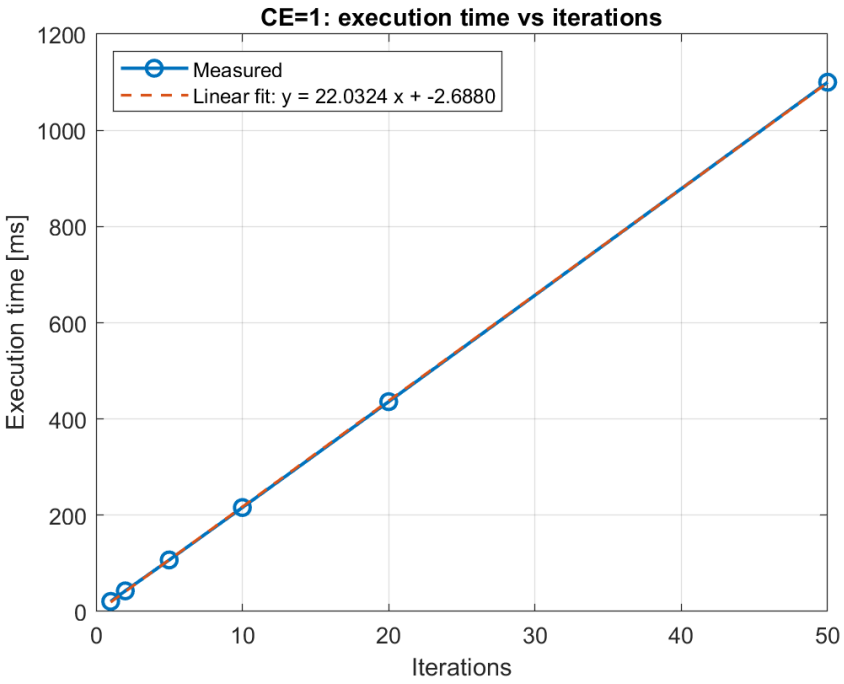


Figure 7.12: Execution time as a function of the number of CoreMark iterations for the control configuration with `CE = '1'`. The BUFGCE-based clock path is present in the design, but the clock is never actually disabled.

The RMS measurements collected for this control configuration are particularly important in the interpretation of the experimental campaign. Since the clock path remains structurally similar to the one used in the gated implementation, but the clock is never actually turned off, the measured idle activity in this case can be used as a reference to determine whether the reduction observed in the gated case is genuinely caused by clock suppression. In fact, the measured idle RMS values remain significantly higher than in the clock-gated case, indicating that a substantial amount of switching activity is still present even when the processor is not performing useful computation.

If the `CE = '1'` configuration exhibits activity levels comparable to the gated case, then the measured benefit could not be safely attributed to effective gating. However, the experimental results show that this is not the case. On the contrary, if the gated implementation shows a lower idle-related activity than the `CE = '1'` case, then the reduction can be interpreted much more confidently as the actual effect of disabling the clock during the sleep phases.

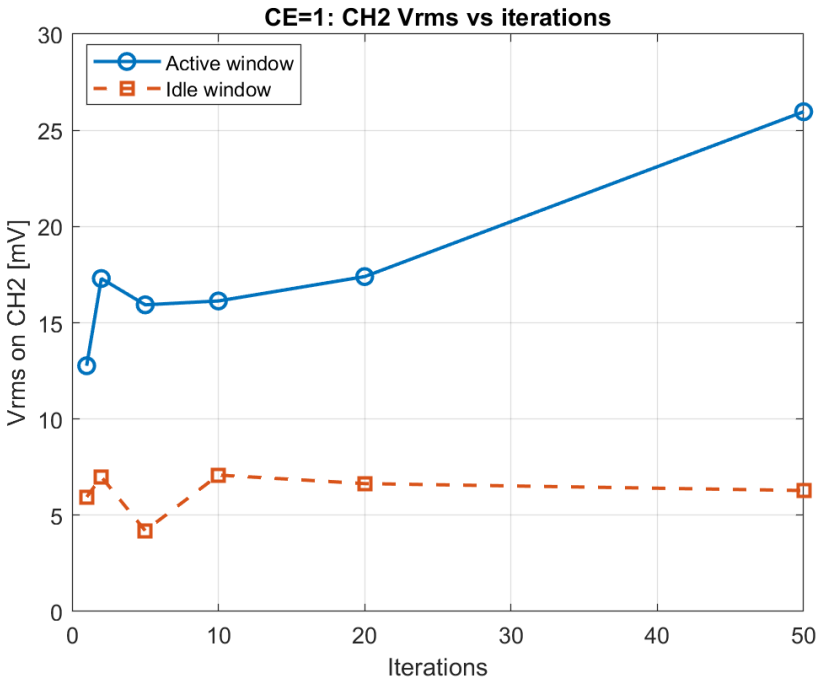


Figure 7.13: Measured active and idle RMS values for the control configuration with `CE = '1'`. These results are used as a reference to distinguish the effect of actual clock suppression from the mere presence of the modified clock path.

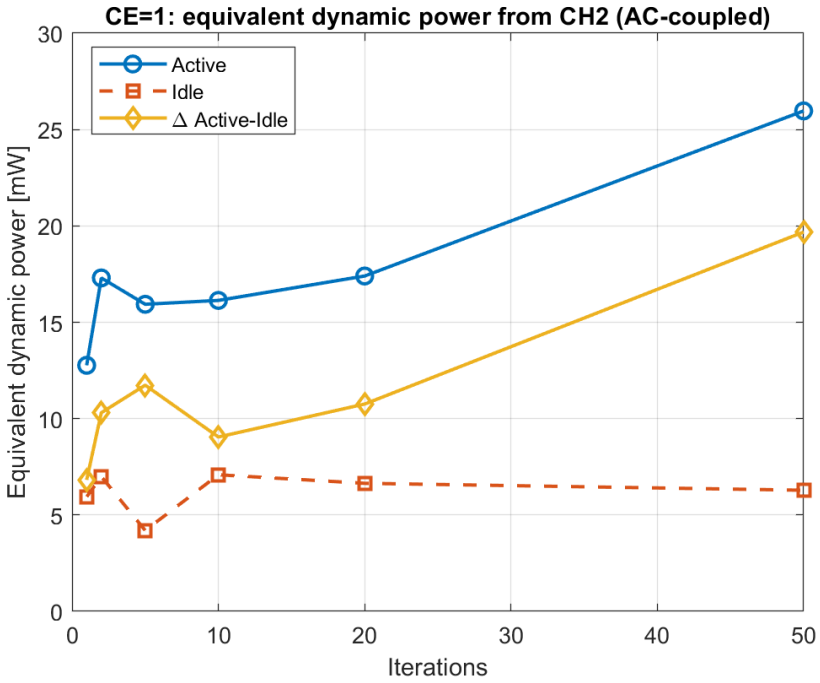


Figure 7.14: Equivalent dynamic power derived from the CH2 RMS measurements for the control configuration with `CE = '1'`. The limited difference between active and idle conditions indicates that software-defined idle intervals alone do not lead to a significant reduction of dynamic activity when compared to the clock-gated configuration when the clock remains continuously enabled.

As expected, the equivalent dynamic power follows the same trend observed in the RMS measurements. In particular, the smaller difference between active and idle conditions compared to the clock-gated case confirms that the absence of clock gating prevents a substantial reduction of switching activity during idle phases. This behavior is consistent with the fact that, in synchronous digital systems, clock activity is one of the dominant contributors to dynamic power consumption. As in the other cases, the quantities derived from the shunt-related waveform should be regarded as equivalent comparative indicators rather than absolute average power values. Their usefulness lies in the fact that all the considered configurations were measured under the same hardware setup, the same oscilloscope conditions, and the same post-processing methodology. This makes the comparison physically meaningful even if the measurements are not intended as precision DC power metrology.

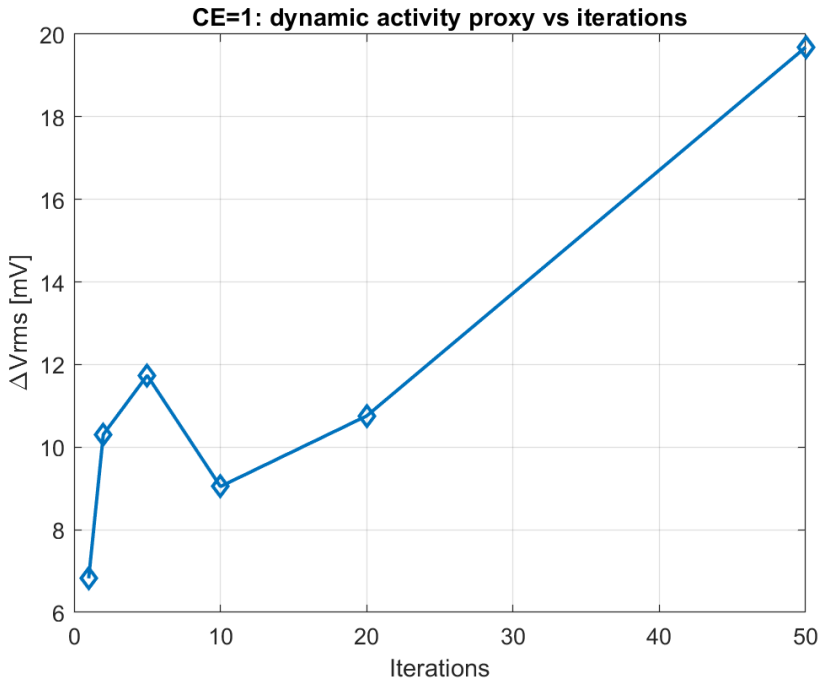


Figure 7.15: Difference between active and idle RMS values for the control configuration with `CE = '1'`. In particular, the relatively limited variation of ΔV_{rms} across iterations confirms that the insertion of idle intervals alone is not sufficient to significantly reduce dynamic activity.

In summary, the `CE = '1'` configuration demonstrates that software-defined idle phases alone do not lead to a meaningful reduction of dynamic activity. Despite the presence of explicit sleep intervals, the continuous propagation of the clock signal maintains a high level of switching activity in the system. This result confirms that effective power reduction requires the actual suppression of the clock, which is only achieved in the clock-gated configuration.

Table 7.3: Experimental results for the control configuration with `CE = '1'`.

Iters	T_{exec} **(ms)**	$V_{rms,A}$ **(mV)**	$V_{rms,I}$ **(mV)**	$P_{eq,A}$ **(mW)**	$P_{eq,I}$ **(mW)**
1	20.6	12.77	5.94	12.77	5.94
2	43.2	17.3	6.99	17.3	6.99
5	108.4	15.93	4.2	15.93	4.2
10	218	16.13	7.09	16.13	7.09
20	436	17.4	6.64	17.4	6.64
50	1,080	25.97	6.28	25.97	6.28

7.1.5 Comparative discussion

The comparison among the considered configurations clearly highlights the distinct contributions of software-defined idle phases and hardware clock gating to the dynamic behavior of the system. The results do not simply show qualitative differences, but reveal measurable trends in both activity and equivalent dynamic power across all configurations. Each configuration has a specific role in the validation flow, and for this reason the discussion should not be limited to a simple baseline-versus-gated comparison.

The baseline case provides the nominal reference for execution time and activity during continuous benchmark execution. Its main role is to validate the trigger-based timing methodology and to establish the reference behavior of the system when no artificial idle phases are introduced. However, precisely because no structured idle interval exists in the baseline case, this configuration alone cannot demonstrate the effect of suppressing clock activity during non-computational phases.

A conceptually important intermediate reference is the sleep-aware baseline, which introduces software-defined idle intervals without applying hardware clock suppression. This configuration clarifies that idle phases alone are not sufficient to significantly reduce dynamic activity when the clock remains active.

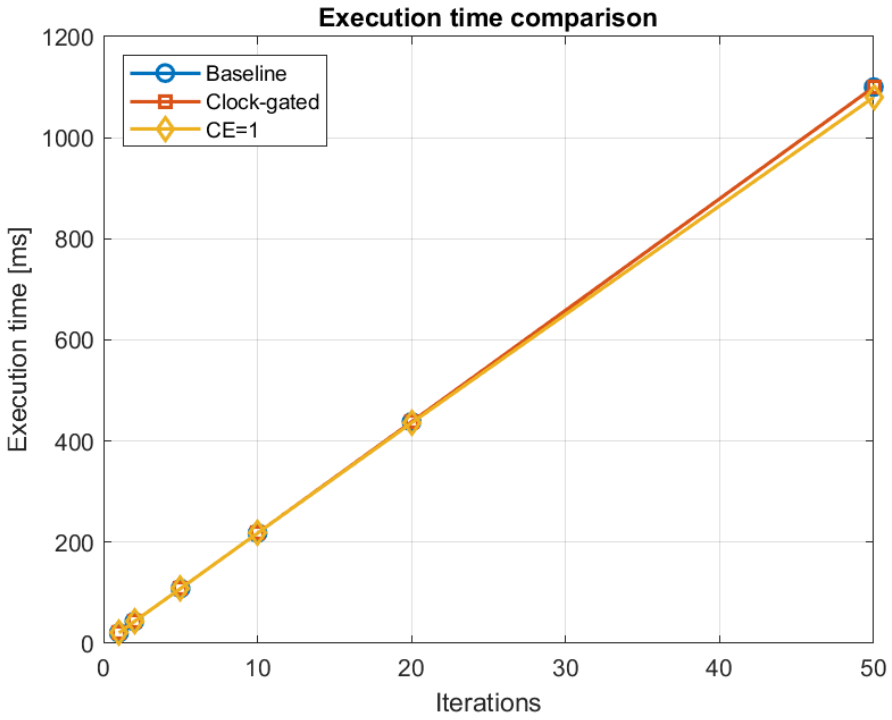


Figure 7.16: Comparison of execution time as a function of the number of CoreMark iterations for the baseline, clock-gated, and `CE = '1'` configurations. The increase observed in the sleep-aware configurations is due to the intentional insertion of idle intervals.

As shown in Fig. 7.16, the execution time of the clock-gated and `CE = '1'` configurations follows an almost identical trend, both exceeding the baseline case due to the intentional insertion of sleep intervals. This confirms that the timing overhead is entirely dominated by the software-defined idle phases, and not by the presence or absence of actual clock gating.

For this reason, the most relevant comparison is between the clock-gated and the `CE = '1'` configurations. These two cases share the same software structure and the same clock-path modification, but differ in a crucial aspect: only the clock-gated implementation actually disables the CPU clock during idle intervals.

This distinction is directly reflected in the measured results. In particular, the idle-related activity observed in the clock-gated case is consistently lower than in the `CE = '1'` configuration, especially at higher iteration counts. This behavior provides strong experimental evidence that the reduction in activity is effectively due to clock suppression, rather than to the presence of idle intervals alone. These two cases share the same software structure, since both include benchmark chunks separated by deliberate sleep intervals. The difference is that, in the clock-gated case, the CPU clock is effectively disabled during the idle intervals, whereas in the non-gated sleep-aware case the processor enters sleep from a software perspective but the clock remains physically active. Therefore, the difference between these two cases directly reflects the contribution of actual hardware clock suppression.

The control configuration with `CE = '1'` further strengthens this interpretation. Since the BUFGCE remains part of the design in both the control and the gated implementations, the comparison between these two cases allows one to separate the effect of real clock disabling from the mere effect of inserting a modified clock path. If the gated case

exhibits reduced idle-related activity with respect to the `CE = '1'` configuration, then the measured improvement can be attributed to the gating action itself rather than to structural side effects in the clock tree.

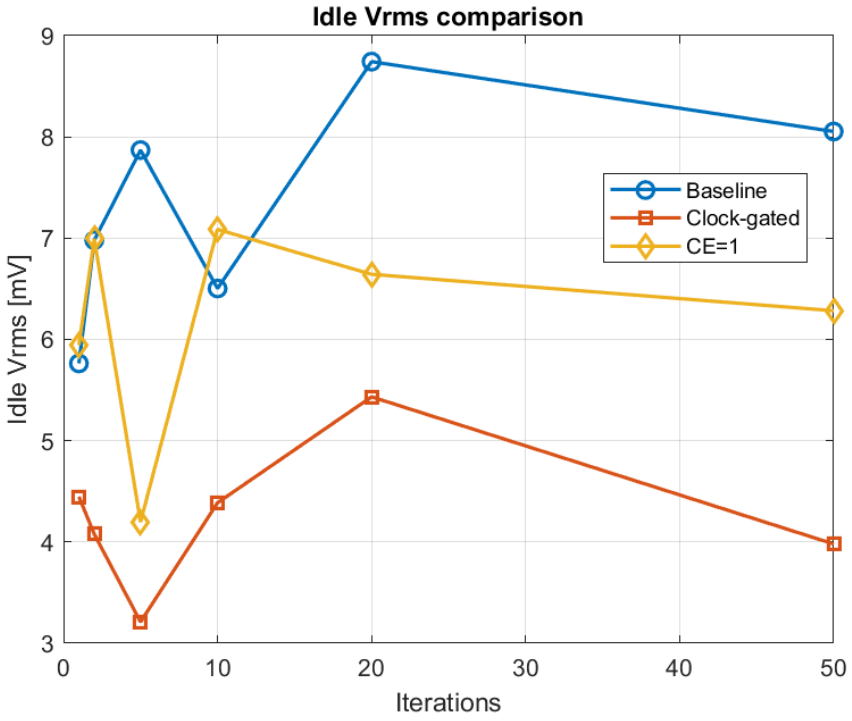


Figure 7.17: Comparison of idle RMS values for the clock-gated and `CE = '1'` configurations. This comparison is particularly important because both designs share the same clock-path structure, while only the former performs actual clock suppression during the sleep phases.

A clear trend emerges from Fig. 7.17:the idle RMS values measured in the `clock-gated` configuration are generally lower than those of the `CE1` control case. This confirms that, although both configurations include identical sleep intervals, only the gated implementation effectively reduces switching activity during these phases.

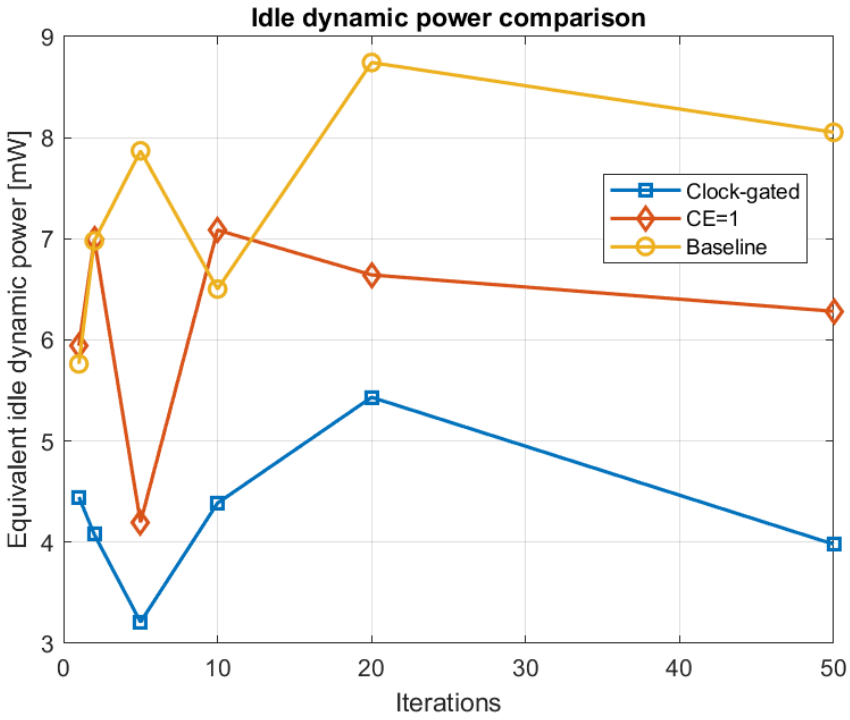


Figure 7.18: Comparison of equivalent idle dynamic power for the considered configurations. The reduction observed in the clock-gated case confirms the effectiveness of hardware clock suppression.

The same trend is observed when considering the equivalent dynamic power derived from the measurements. The clock-gated configuration generally shows lower idle power than the `CE = '1'` case, especially at several of the higher-significance operating points. However, the magnitude of the reduction is not uniform across all iteration counts, reflecting the variability of the measured waveforms and the sensitivity of the RMS-based estimation to the observation window. It is also worth noting that the absolute reduction tends to increase at higher iteration counts, where longer observation windows provide a more stable estimate of the activity difference. This suggests that the effectiveness of clock gating becomes more evident under sustained workloads, where idle intervals accumulate over longer execution times.

Another important aspect concerns the interpretation of the RMS measurements. Because the Channel 2 acquisitions were often performed in AC coupling mode, the reported current- and power-related quantities should not be read as exact absolute power consumption values of the full FPGA system. Nevertheless, they remain meaningful comparative indicators of dynamic activity, especially when all the measurements are collected under identical supply, routing, and acquisition conditions. In this context, the comparison of active and idle values across the different configurations is more relevant than the absolute magnitude of any single measured point.

The results also show that the trigger-based approach provides a robust practical framework for correlating timing and activity. The trigger waveform makes it possible to identify the measured benchmark section unambiguously, while the simultaneous observation of the shunt-related signal allows active and idle phases to be visually and quantitatively distinguished. This dual observation is essential for validating the clock-gating strategy in an experimentally transparent way.

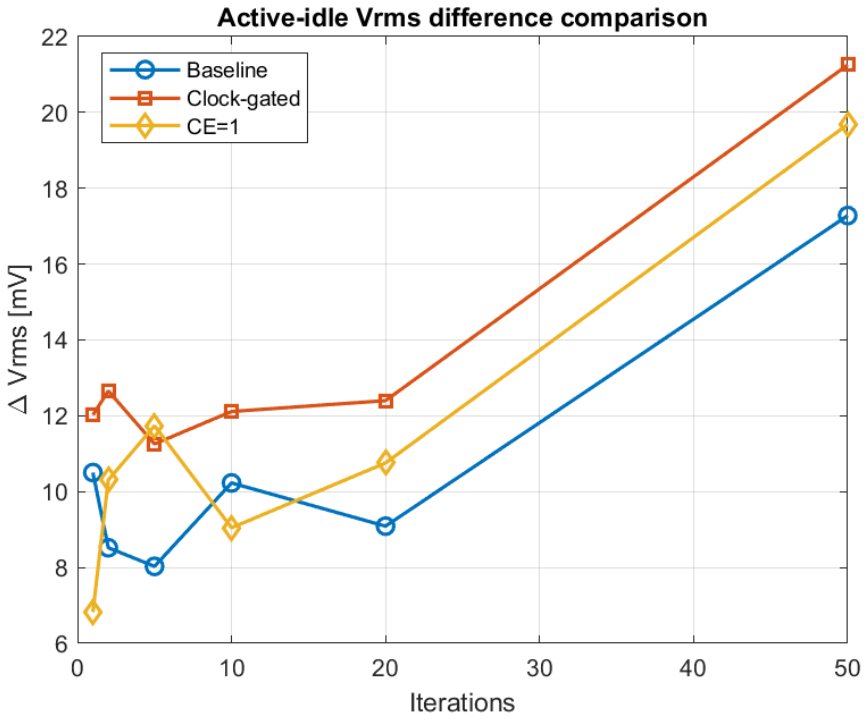


Figure 7.19: Comparison of the active-idle RMS difference for the considered configurations. This quantity provides a compact indicator of the relative activity variation between computation and idle intervals.

The comparison of the active-idle difference further strengthens this conclusion. As shown in Fig. 7.19, the clock-gated and `CE = '1'` configurations exhibit a significantly larger separation between active and idle conditions compared to the baseline case. This indicates that the introduction of structured idle intervals already creates a measurable activity contrast, but only the gated implementation translates this contrast into an actual reduction of idle activity. It is also worth noting that some variability is present in the low-iteration measurements, which can be attributed to acquisition noise and the limited duration of the observation window. However, the overall trends remain consistent across all configurations and become increasingly clear as the number of iterations grows.

Overall, the experimental results provide clear evidence that the introduction of software-defined idle phases alone is not sufficient to reduce dynamic activity. Only when these idle intervals are combined with actual hardware clock gating does a measurable reduction in switching activity occur. The `CE = '1'` control configuration plays a crucial role in this validation, as it demonstrates that the observed improvements are not due to structural modifications of the clock tree, but are directly caused by effective clock suppression.

Table 7.4: Comparison of equivalent idle dynamic power for the considered configurations.

Iters	**Base. (mW)**	**Gated (mW)**	**CE=1 (mW)**	ΔP_{idle} **(mW)**	**Saving (%)**
1	5.76	4.45	5.94	1.5	25.18
2	6.98	4.08	6.99	2.92	41.71
5	7.87	3.21	4.2	0.99	23.51
10	6.5	4.39	7.09	2.7	38.1
20	8.74	5.43	6.64	1.21	18.21
50	8.05	3.98	6.28	2.3	36.66

The quantity ΔP_{idle} represents the difference between the idle equivalent dynamic power measured in the `CE = '1'` configuration and in the clock-gated implementation. Positive values therefore indicate a reduction in idle activity achieved through clock gating.

The percentage saving is computed with respect to the `CE = '1'` configuration as

$$\text{Saving}_{CE=1} = \frac{P_{\text{idle,CE1}} - P_{\text{idle,gated}}}{P_{\text{idle,CE1}}} \cdot 100.$$

Positive values indicate that the clock-gated configuration reduces the equivalent idle dynamic power with respect to the permanently enabled clock-path case.

In all the considered operating points, ΔP_{idle} remains strictly positive, indicating that the clock-gated configuration consistently achieves lower idle-related dynamic activity than the `CE = '1'` case.

Although the magnitude of the reduction is not uniform across all iteration counts, the absence of negative values confirms that the observed benefit is robust and not affected by measurement variability. This provides strong experimental evidence that the reduction in idle activity is effectively due to clock suppression.

7.2 Arty A7-100T XADC-Based Results

The second part of the experimental analysis was performed on the Arty A7-100T platform using the on-board current-sense path and the FPGA XADC. Unlike the CW305 campaign, which relied on oscilloscope-based RMS extraction, this setup collects phase-labelled samples directly in hardware and exports them through UART for MATLAB post-processing. The objective is to compare the same conceptual configurations with a more automated and phase-aware acquisition flow.

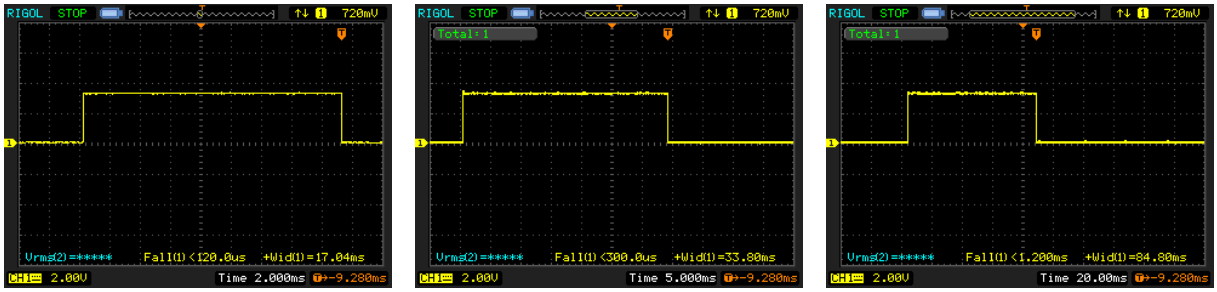
7.2.1 Baseline Characterization on Arty A7-100T (BASE)

The first experimental activity focused on the baseline characterization of the NEORV32 platform implemented on the Arty A7-100T FPGA board. The objective was to evaluate execution-time behavior and idle-power stability while running the CoreMark benchmark with different iteration counts.

The processor was configured to operate at 100 MHz, and the execution interval was externally measured using a GPIO-based trigger connected to the oscilloscope. The trigger

signal was asserted immediately before benchmark execution and deasserted immediately afterward, allowing direct observation of the active computation window.

Figure 7.20 shows three representative oscilloscope acquisitions corresponding to 1, 2, and 5 benchmark iterations. Channel 1 (yellow) represents the trigger signal generated by the GPIO instrumentation. The pulse width increases proportionally with the number of iterations, confirming deterministic benchmark execution and correct trigger synchronization.



(a) Single iteration. (b) Two iterations. (c) Five iterations.

Figure 7.20: Oscilloscope acquisitions for the baseline configuration using different CoreMark iteration counts. The trigger pulse duration scales proportionally with the workload, confirming deterministic execution behavior.

The measured execution times extracted from the oscilloscope acquisitions were approximately:

- 17 ms for 1 iteration,
- 34 ms for 2 iterations,
- 85 ms for 5 iterations.

The observed scaling is nearly linear with the iteration count, indicating that the benchmark workload behaves deterministically and that no significant timing jitter or asynchronous disturbances affect execution.

In parallel with the external timing measurements, the FPGA VCCINT current consumption was monitored through the onboard current-sense path connected to the XADC subsystem. For this purpose, the CoreMark main program was extended with a set of dedicated measurement functions. These functions read the raw XADC conversion value exposed through the GPIO input port, convert it into an estimated current value, and print the samples through UART in CSV format. Each sample is therefore stored as a tuple containing the acquisition phase, the sample index, the raw ADC value, and the corresponding current expressed in microamperes.

More specifically, the acquisition routine was structured around two idle measurement windows. The first one, denoted as `idle_before`, is collected before the benchmark execution starts. The second one, denoted as `idle_after`, is collected immediately after the benchmark execution has completed. In the present baseline characterization, 100 samples were acquired in each of these two windows. Therefore, each run provides 100 samples describing the idle state before the workload and 100 samples describing the idle state after the workload.

This before–after structure was introduced for two reasons. First, the `idle_before` samples provide a reference value for the steady-state consumption of the FPGA before the processor enters the measured computation interval. Second, the `idle_after` samples make it possible to verify whether the execution of the benchmark produces any persistent change in the idle operating point, for example due to thermal drift, residual switching activity, or measurement instability. If the two sets of samples remain close to each other, the idle power estimate can be considered stable and the average of the two windows can be used as a representative idle value for that run.

The UART output generated by the firmware was then saved as a text log and processed offline in MATLAB. During post-processing, only the CSV lines corresponding to `idle_before` and `idle_after` were retained. The raw XADC values were converted into current using the Arty A7 sensing relationship, and the corresponding VCCINT power was obtained by multiplying the estimated current by the nominal core voltage. For each iteration count, the MATLAB script computed the mean and standard deviation of the idle current and idle power separately for the before and after windows, and then combined the two sets into a single average idle estimate.

In this baseline configuration, no active current samples were collected during the benchmark execution itself. The oscilloscope trigger was used only to measure the computation time, while the XADC-based data stream was used to characterize the idle condition around the benchmark execution. This choice was intentional: the purpose of the baseline experiment was not yet to resolve fine-grained active/idle transitions, but to establish a stable reference level for the FPGA core consumption before moving to the clock-gated configurations.

The measured idle power, obtained by combining the 100 `idle_before` samples and the 100 `idle_after` samples of each run, remained remarkably stable across all tested iteration counts.Figure 7.21 reports the measured idle power before execution, after execution, and the combined average value.

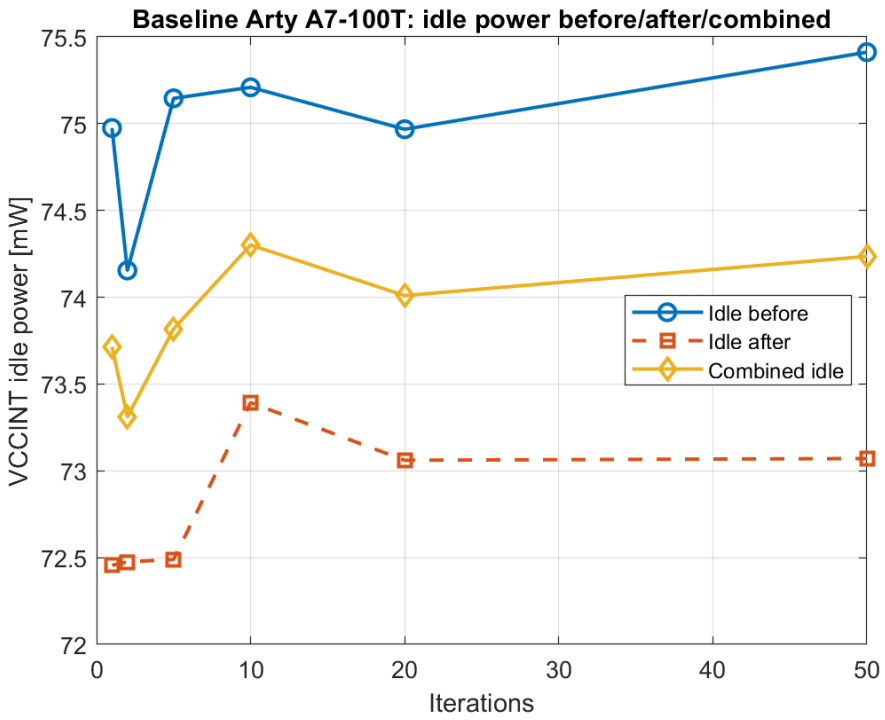


Figure 7.21: Measured idle V_{CCINT} power before and after CoreMark execution for different iteration counts. The combined idle value corresponds to the average of the two measurement windows. Error bars are not shown in this figure; the standard deviation from repeated acquisitions is reported separately in Fig. 7.22.

The average idle power remains within approximately 73 mW to 75 mW over the entire iteration range. The difference between the *idle before* and *idle after* measurements is consistently limited, typically below 2 mW. This indicates that benchmark execution does not introduce significant thermal drift or persistent activity variations in the FPGA fabric during the considered time intervals.

No error bars are shown in Fig. 7.21, because this figure is intended to compare the two idle windows and their combined average. The dispersion obtained from repeated XADC acquisitions is reported separately in Fig. 7.22.

The largest point-to-point variations are observed at some individual operating points. A different number of XADC samples does not cause these fluctuations, since the acquisition length is fixed. They are more likely related to the intrinsic variability of the XADC conversion process, quantization effects, and small changes in the board operating condition between repeated runs. For larger workloads, the measurements become more stable and the average idle value converges toward a nearly constant level.

Figure 7.22 reports the mean idle power together with the associated standard deviation obtained from repeated XADC acquisitions.

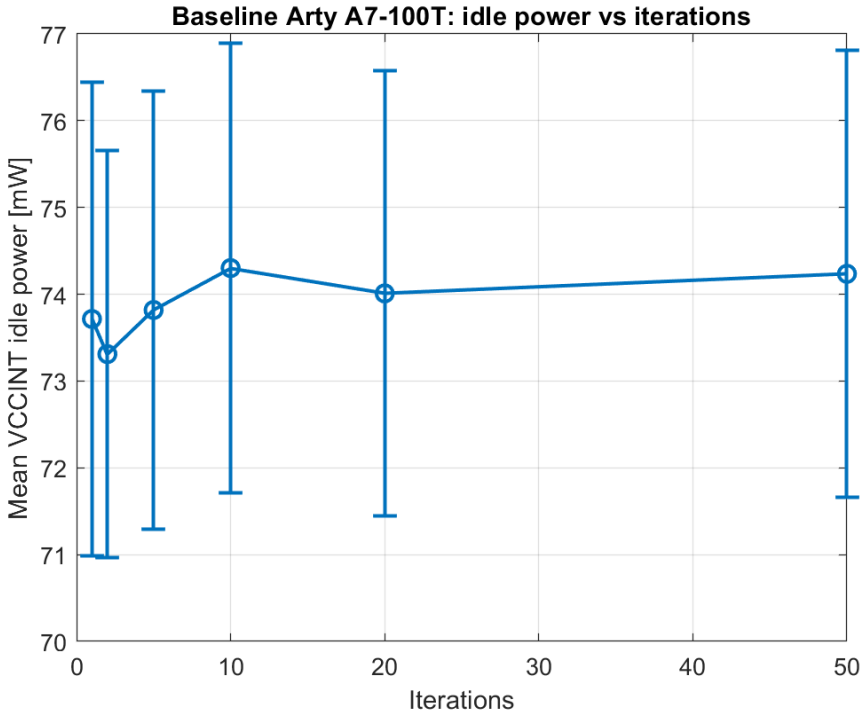


Figure 7.22: Average idle V_{CCINT} power with corresponding standard deviation for different benchmark iteration counts, obtained from repeated XADC acquisitions.

The standard deviation remains limited over all operating points, confirming the repeatability of the acquisition methodology and the stability of the experimental setup. Although small fluctuations are visible, no progressive increase in idle consumption is observed as the benchmark duration increases.

Table 7.5 summarizes the processed experimental results extracted from the XADC acquisitions.

Table 7.5: Experimental idle-power characterization results for the baseline configuration.

Iterations	$P_{idle,before}$ **(mW)**	$P_{idle,after}$ **(mW)**	$P_{idle,avg}$ **(mW)**	σ_{idle} **(mW)**
1	74.98	72.46	73.72	2.73
2	74.15	72.48	73.31	2.35
5	75.15	72.49	73.82	2.52
10	75.21	73.39	74.3	2.59
20	74.97	73.06	74.01	2.56
50	75.41	73.07	74.24	2.58

Overall, the baseline characterization demonstrates three important properties of the platform:

1. deterministic execution timing;
2. stable idle-power behavior;
3. reliable trigger-based synchronization for external measurements.

These results establish a solid reference point for the subsequent experiments involving clock-gating and power-management techniques. In particular, the observed stability of the idle operating condition is important because it allows subsequent reductions in activity to be attributed to the proposed hardware gating strategy rather than to intrinsic fluctuations of the measurement setup.

After the baseline characterization presented in the previous section, the Arty A7-100T measurement flow was extended to configurations in which the CoreMark execution is explicitly divided into active computation phases and low-activity intervals. The purpose of this second Arty-based campaign was to verify whether the power reduction observed during sleep intervals is caused only by the software execution of the `WFI` instruction or by the actual suppression of the CPU clock through the BUFGCE-based clock-gating structure.

Unlike the first baseline experiment, where only `idle_before` and `idle_after` samples were collected, the updated firmware records samples during four different execution phases:

- `idle_before`: idle measurement window before the benchmark execution;
- `active`: CoreMark computation window;
- `sleep`: software-defined sleep interval between computation chunks;
- `idle_after`: idle measurement window after benchmark execution.

This phase-aware acquisition makes it possible to compare the measured VCCINT activity during useful computation and during the inserted low-power intervals. However, to make this comparison meaningful, the acquisition system had to be modified so that phase labels were associated with the samples at the time of acquisition rather than at the time of software readout.

7.2.2 Phase-Aware XADC Acquisition

The first implementation of the XADC measurement path stored only the 12-bit raw ADC sample inside the hardware FIFO. The software then assigned the phase label when the FIFO was dumped through UART. This approach was not sufficiently robust, because the XADC continued to acquire samples while the CPU was printing data. As a result, the samples read by the software could be temporally shifted with respect to the phase label assigned during the dump.

To remove this ambiguity, the FIFO entry was extended to include both the raw XADC sample and a two-bit phase code sampled at acquisition time. The logical format of each FIFO entry is therefore:

$$\text{FIFO entry} = \{\text{phase}[1:0], \text{raw}[11:0]\}\,.$$

The phase code uses the following encoding:

Phase code	Meaning
00	`idle_before`
01	`active`
10	`sleep`
11	`idle_after`

A dedicated capture-enable signal was also introduced. When this signal is low, the XADC monitor remains clocked but no new sample is written into the FIFO. This prevents the FIFO from being filled during UART transmission and ensures that only samples belonging to the selected acquisition window are stored.

The resulting measurement sequence for each phase is:

1. disable capture;
2. flush the FIFO;
3. set the hardware phase code;
4. enable capture;
5. execute the target phase;
6. disable capture;
7. dump the FIFO through UART.

This modification is essential for the interpretation of the results. Each sample now carries the phase information captured at the same time as the raw XADC value. Therefore, the `active` and `sleep` datasets are no longer affected by software-side labelling offsets or UART-induced FIFO contamination.

Based on the phase-labelled datasets produced by this acquisition flow, the following quantities are used in the subsequent tables. P_{active} denotes the mean reconstructed V_{CCINT} power during the active CoreMark computation phase, while P_{sleep} denotes the mean reconstructed power during the complete software-defined sleep window. The corresponding standard deviations are denoted by σ_{active} and σ_{sleep}, respectively. Finally, $\Delta P_{\text{active}-\text{sleep}}$ denotes the difference between the active and sleep mean power values:

$$\Delta P_{\text{active}-\text{sleep}} = P_{\text{active}} - P_{\text{sleep}}.$$

For each phase-aware configuration containing a software-defined low-activity interval, two low-phase metrics are considered. $P_{\text{low,total}}$ denotes the mean reconstructed V_{CCINT} power over the complete low-activity window, including the initial active-to-low transition. $P_{\text{low,steady}}$ denotes the mean power after excluding all low-phase samples with phase-local index lower than 75. This separates the transition region from the settled low-activity behavior and allows CGATE, SLEEP, and CE1 to be compared using the same post-processing rule.

7.2.3 Clock-Gated Configuration (CGATE)

The clock-gated configuration represents the actual low-power implementation evaluated on the Arty A7-100T platform. In this configuration, the CoreMark benchmark is executed in chunks, and a timer-based sleep interval is inserted between consecutive computation windows. During the active phase, the processor executes CoreMark instructions. During the sleep phase, the software programs the machine timer compare register and executes the RISC-V `WFI` instruction through `neorv32_cpu_sleep()`.

At hardware level, the CPU clock is generated through a BUFGCE primitive. The clock enable is deasserted when the processor enters the sleep state and is reasserted when

a wake-up source becomes pending. In the implemented design, the CPU clock is kept enabled during reset, while the CPU is awake, or whenever one of the configured wake-up sources is pending. The resulting enable condition is shown in Listing 7.1.

```
-- Clock ON when CPU is not sleeping, during reset, or on wake-up
   events
ce_cpu <= (not rstn_sys) or
          (not cpu_sleep(0)) or
          mti(0) or
          msi(0) or
          irq_mei_i or
          or_reduce_f(cpu_firq);
```

Listing 7.1: CPU clock-enable condition used for the gated CPU clock domain.

For this experiment, the XADC-based acquisition was performed using the phase-aware FIFO mechanism described previously. Each XADC sample was stored together with the corresponding hardware phase code, so that the phase label used during MATLAB post-processing reflects the phase in which the sample was actually acquired. This avoids the ambiguity that would arise if the software assigned the phase only when dumping the FIFO through UART.

The processed UART logs show a clear and repeatable separation between active and sleep phases. For the clock-gated configuration, P_{active} remains close to approximately 85 mW, while P_{sleep} drops to approximately 50 mW when the complete sleep window is considered. In this analysis, all sleep samples are included in the computation of P_{sleep}, including the initial transition from active execution to the lower-activity sleep state.

Table 7.6: Arty A7-100T XADC results for the clock-gated configuration using the complete sleep window, including the initial active-to-sleep transition.

Iterations	P_{active} (mW)	σ_{active} (mW)	P_{sleep} (mW)	σ_{sleep} (mW)	$\Delta P_{\text{active}-\text{sleep}}$ (mW)
1	85.35	2.19	50.71	10.82	34.64
2	85.46	2.11	50.74	10.76	34.71
5	85.35	2.11	50.59	10.71	34.76
10	85.28	2.04	50.55	10.73	34.74
20	85.28	2.10	50.36	10.71	34.92
50	85.31	2.14	50.58	10.79	34.73

The standard deviation of the complete sleep window is significantly higher than the standard deviation of the active phase because the sleep dataset includes the complete transition from the active state to the clock-gated low-activity state. The first samples of each sleep interval still reflect residual activity immediately after the execution of `WFI`, while the following samples correspond to the lower activity level reached after the CPU clock has been suppressed. Therefore, the reported P_{sleep} value in Table 7.6 should be interpreted as the average over the complete sleep window rather than as a pure steady-state value.

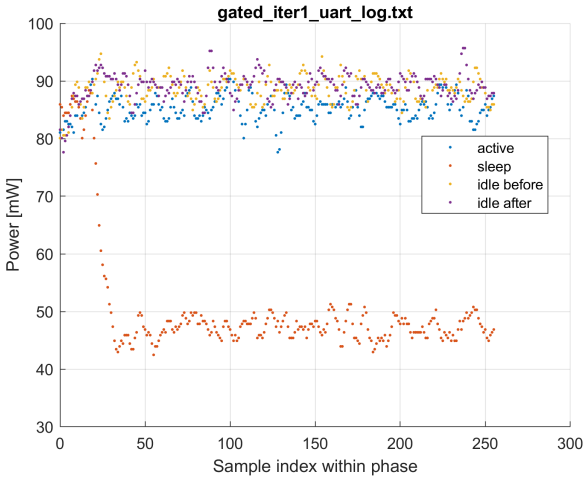
To separate the mode-switching transient from the settled low-activity region, an additional steady-state sleep metric was computed by excluding all sleep samples with phase-local index lower than 75. The resulting values are reported in Table 7.7.

Table 7.7: Steady-state sleep statistics for the clock-gated configuration after excluding the first 75 sleep samples.

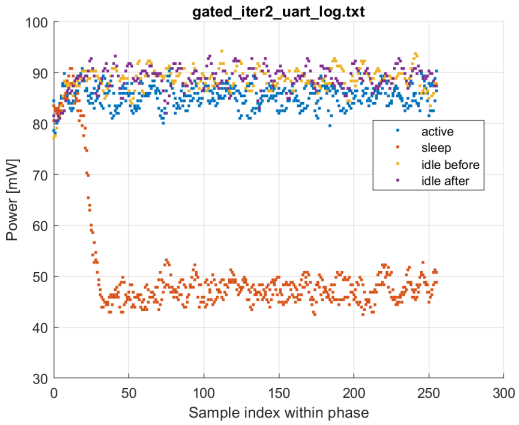
Iterations	$P_{\text{sleep,steady}}$ (mW)	$\sigma_{\text{sleep,steady}}$ (mW)	$\Delta P_{\text{active-sleep,steady}}$ (mW)
1	47.33	1.78	38.02
2	47.45	2.06	38.01
5	47.28	1.93	38.08
10	47.17	1.96	38.11
20	46.98	1.96	38.30
50	47.14	1.96	38.17

After excluding the initial transition samples, the sleep standard deviation decreases from approximately 10.7 mW to about 2 mW. This confirms that the large standard deviation of the complete sleep window is mainly caused by the active-to-sleep transition, rather than by instability of the clock-gated low-power state. The steady-state sleep power remains close to 47 mW, corresponding to an active-to-steady-sleep difference of approximately 38 mW.

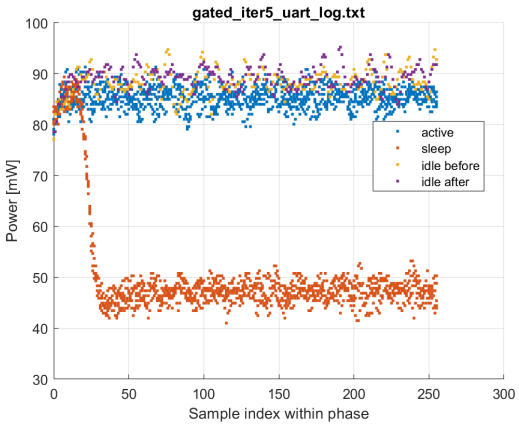
Figure 7.23 shows a representative phase-wise distribution of the XADC samples. The active samples remain clustered around the higher-power region, whereas the sleep samples show a visible drop toward a lower-power region after the initial transition. This behavior confirms that the BUFGCE-based clock gating effectively reduces the switching activity of the CPU clock domain during the software-defined sleep intervals.



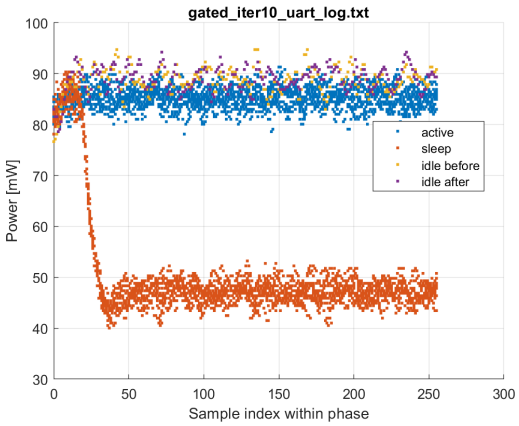
(a) 1 CoreMark iteration.



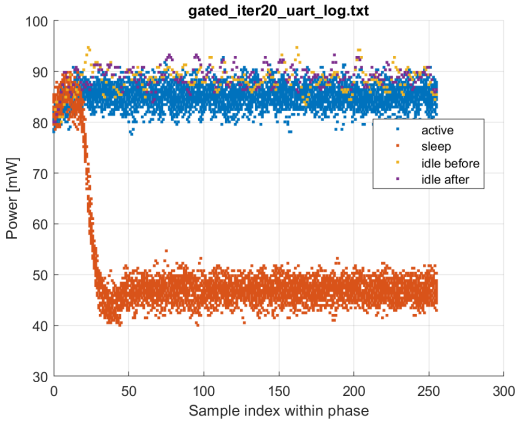
(b) 2 CoreMark iterations.



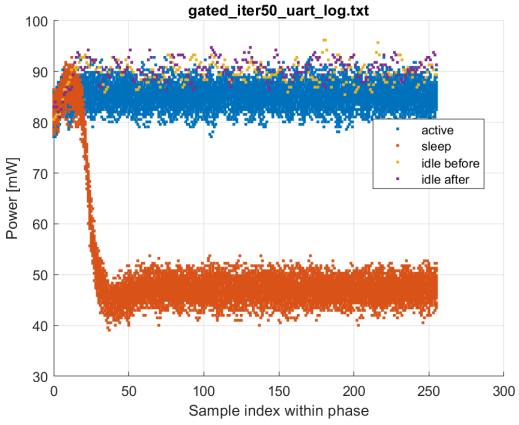
(c) 5 CoreMark iterations.



(d) 10 CoreMark iterations.



(e) 20 CoreMark iterations.



(f) 50 CoreMark iterations.

Figure 7.23: Phase-wise XADC power samples for the clock-gated configuration on the Arty A7-100T platform. Each panel reports the samples acquired during the `idle_before`, `active`, `sleep`, and `idle_after` phases for a different number of CoreMark iterations. The horizontal axis represents the sample index within each phase, while the vertical axis reports the reconstructed VCCINT power from the XADC raw values.

Figure 7.23 reports the phase-wise distribution of the XADC samples collected in the clock-gated configuration. The purpose of representing the six iteration counts in a single figure block is to emphasize the repeatability of the measured behavior rather than the details of a single run. All panels use the same type of axes and the same phase encoding, so that the evolution of the active and sleep clusters can be compared directly across different benchmark lengths.

The `active` samples correspond to the intervals in which the NEORV32 processor executes the CoreMark workload. These samples remain clustered in the upper region of the plots, typically around 80–90 mW. The `sleep` samples correspond to the timer-controlled intervals in which the processor executes `WFI` and the BUFGCE clock enable suppresses the CPU clock. In all the considered runs, the sleep samples show a clear transition from the active-power region toward a lower-power region, typically around 45–50 mW after the initial transient.

The first sleep samples are still close to the active level because they include the transition immediately after the execution of `WFI`. After this short transient, the samples settle to a lower and comparatively stable power level. This behavior is visible for all iteration counts and confirms that the measured reduction is not caused by an isolated acquisition artifact, but by the repeated transition into the clock-gated sleep state.

As the number of iterations increases, the number of active and sleep samples also increases, making the point clouds denser. This is especially visible in the 20- and 50-iteration cases. Nevertheless, the separation between the active cluster and the low-power sleep cluster remains clearly visible, confirming that the observed behavior does not depend on a single short acquisition window.

The `idle_before` and `idle_after` phases are also reported as reference conditions. These phases should not be interpreted as low-power sleep states, because the processor is still active from a software point of view. Their role is instead to verify the stability of the operating point before and after benchmark execution. The fact that these samples remain in the same high-power region as the active phase confirms that the large reduction observed during `sleep` is specifically associated with the clock-gated WFI interval.

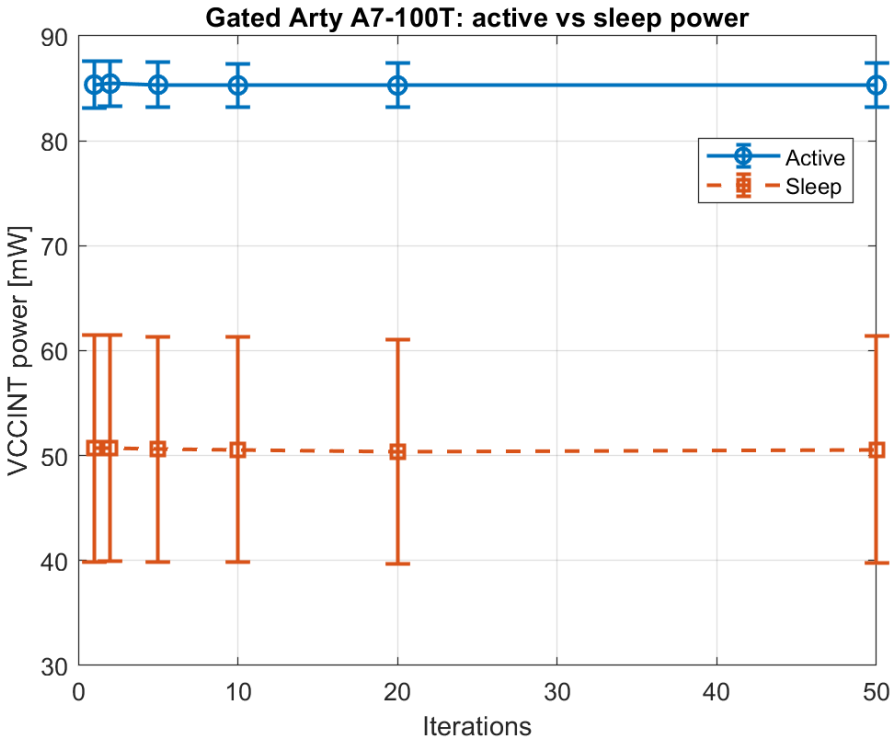


Figure 7.24: Average active, complete-sleep, and steady-state sleep V_{CCINT} power for the clock-gated configuration. Error bars represent the standard deviation of the XADC samples acquired in each phase. The steady-state sleep values are computed after excluding the first 75 samples of each sleep interval. Lines are drawn only as visual guides between the tested iteration counts.

Figure 7.24 summarizes the average reconstructed VCCINT power for the active and sleep phases. The active power remains approximately constant around 85 mW, while the sleep power remains around 50 mW over the complete iteration range. The separation between the two curves confirms that the clock-gated sleep intervals produce a repeatable reduction in measured activity.

The larger error bars associated with the sleep phase are expected, since the sleep samples include both the initial transition after the execution of `WFI` and the subsequent lower-activity region. Therefore, the sleep standard deviation should not be interpreted only as random measurement noise, but also as the effect of the transition from active execution to the gated low-power state.

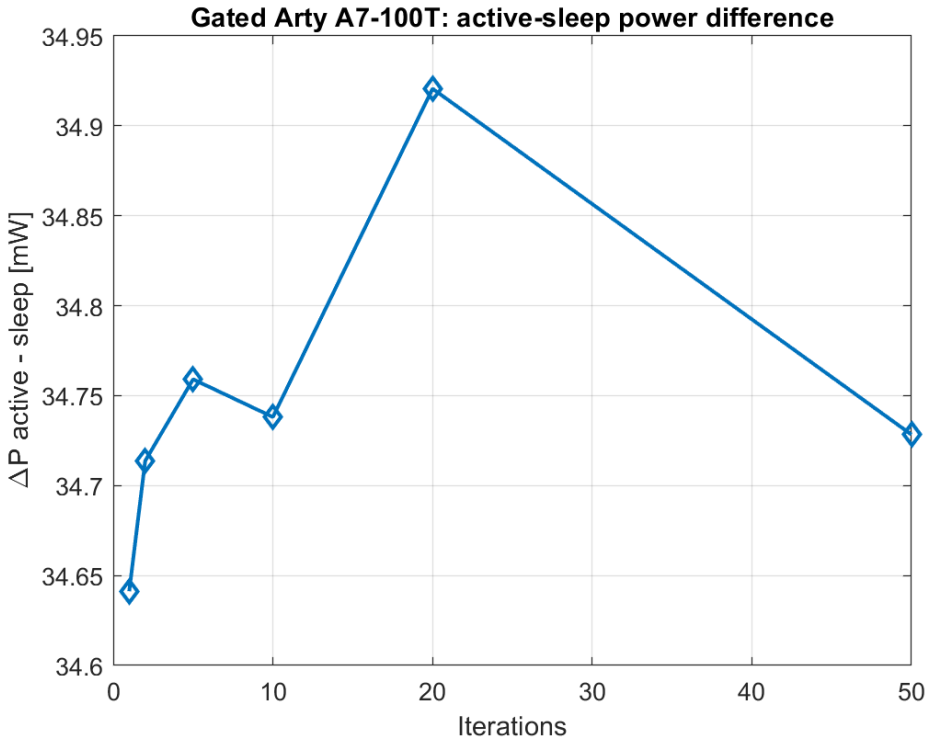


Figure 7.25: Measured active-sleep power difference for the clock-gated configuration.

Figure 7.25 reports the active-sleep power difference computed from the phase-wise XADC samples. The measured reduction remains close to 35 mW for all tested iteration counts. This stability is important because it shows that the measured benefit is not an isolated artifact of a single short run. Instead, the reduction is consistently observed as the number of CoreMark iterations increases.

The limited variation of $\Delta P_{\text{active}-\text{sleep}}$ also confirms that the phase-aware FIFO acquisition and the capture-enable mechanism provide a stable measurement flow. Since each sample is labelled in hardware at acquisition time, the computed difference reflects the actual separation between active execution and gated sleep intervals rather than a software-side labelling offset. Overall, the clock-gated measurements demonstrate that the proposed BUFGCE-based clock suppression produces a consistent and measurable reduction of the VCCINT activity during sleep intervals. Although the reported sleep average includes the transition phase, the reduction remains large and stable, making the clock-gated configuration clearly distinguishable from the active execution state.

7.2.4 Sleep-aware configuration (SLEEP)

The Sleep-aware configuration (SLEEP) was introduced as an intermediate reference between the pure baseline execution and the fully clock-gated implementation. This configuration uses the same phase-aware XADC acquisition infrastructure adopted for the gated case, including the hardware FIFO, the sampled phase code, and the capture-enable mechanism. It also uses the same software structure based on active CoreMark chunks separated by `WFI`-based sleep intervals. However, in this configuration the processor clock is not effectively suppressed during the sleep phase.

This configuration is useful because it separates the effect of the software sleep mechanism from the effect of true hardware clock gating. In other words, the processor is

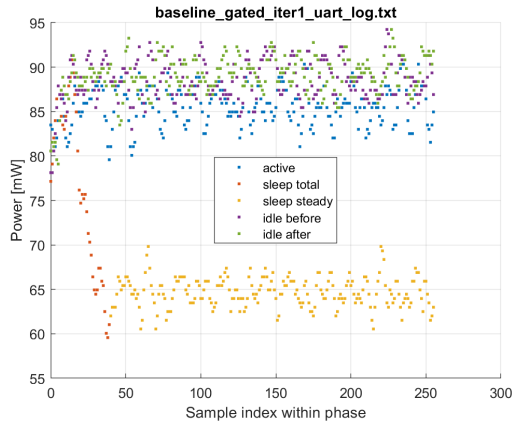
allowed to enter a sleep state from the software point of view, but the clock-driven part of the system does not reach the same low-activity condition observed when the BUFGCE actually disables the CPU clock domain.

In the MATLAB post-processing, five phase groups are represented:

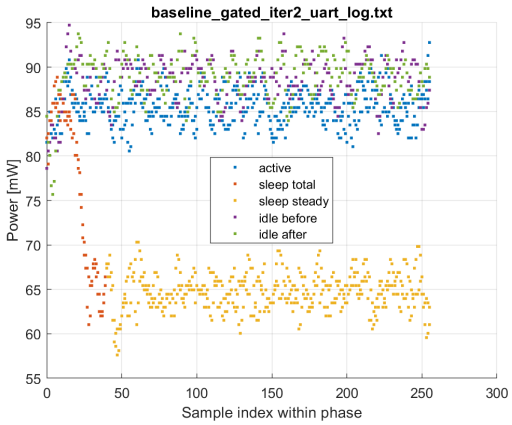
- `active`: samples acquired while the processor executes CoreMark chunks;
- `sleep total`: all samples acquired during the `WFI`-based sleep interval, including the initial transition;
- `sleep steady`: the subset of sleep samples after the initial transient, used to represent the settled sleep level;
- `idle before`: samples acquired before benchmark execution;
- `idle after`: samples acquired after benchmark execution.

The distinction between `sleep total` and `sleep steady` is important because the sleep interval is not instantaneous. The first samples acquired after entering `WFI` still belong to the transition from the active operating point toward the lower-activity sleep condition. The `sleep total` group therefore describes the complete sleep interval, while the `sleep steady` group highlights the settled low-activity level after the transition. This is why five groups are shown in the following phase-wise plots, even though the firmware phases are conceptually based on active, sleep, and idle reference windows.

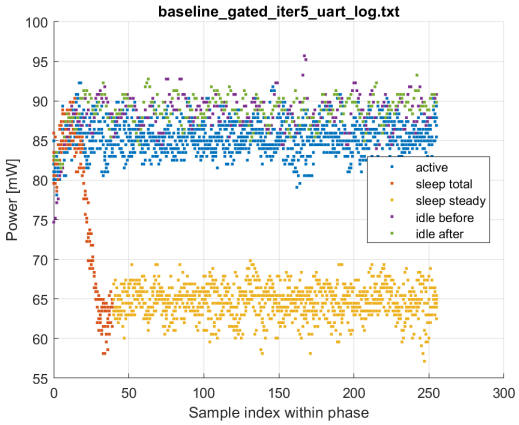
Figure 7.26 reports the phase-wise XADC samples for different CoreMark iteration counts. The active samples remain clustered in the upper region of the plots, typically around 82–89 mW. The sleep samples initially start close to the active level and then move toward a lower level. However, the steady sleep region remains around 63–66 mW, which is significantly higher than the fully clock-gated case. This confirms that `WFI` alone reduces the activity of the processor but does not eliminate the residual clock-driven switching.



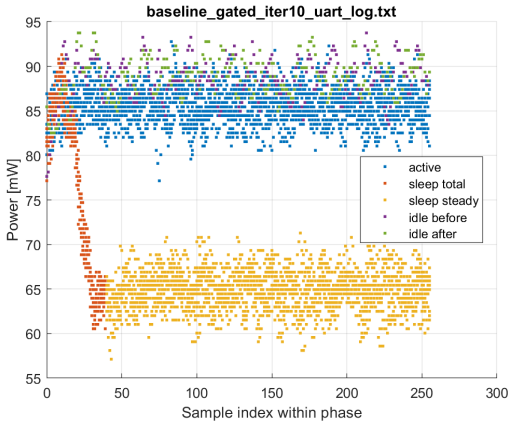
(a) 1 CoreMark iteration.



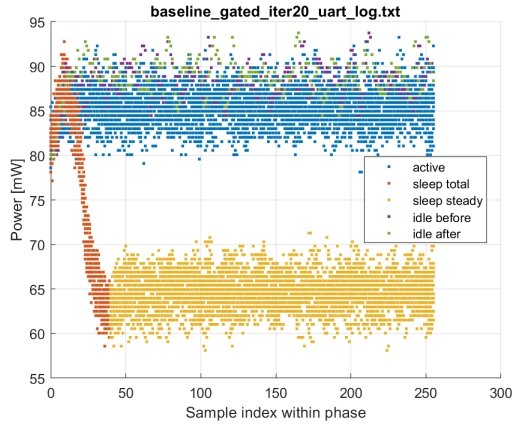
(b) 2 CoreMark iterations.



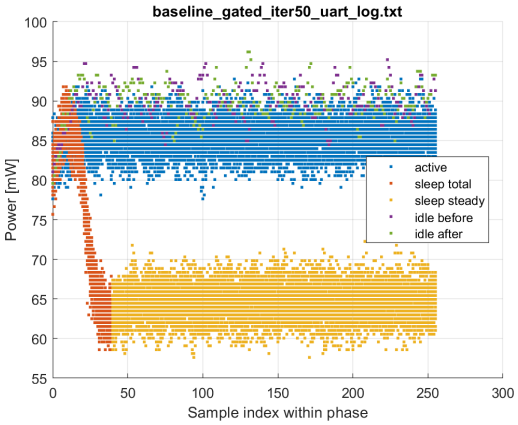
(c) 5 CoreMark iterations.



(d) 10 CoreMark iterations.



(e) 20 CoreMark iterations.



(f) 50 CoreMark iterations.

Figure 7.26: Phase-wise XADC power samples for the Sleep-aware configuration (SLEEP) on the Arty A7-100T platform. Five groups are shown: `active`, `sleep total`, `sleep steady`, `idle before`, and `idle after`. The additional `sleep steady` group is obtained during post-processing to distinguish the settled sleep level from the initial transition included in the complete sleep window.

The six panels in Fig. 7.26 are represented together to show that the behavior is repeatable as the number of CoreMark iterations increases. For low iteration counts, the

number of samples is smaller and the transition region is more visually evident. For higher iteration counts, especially 20 and 50 iterations, the point clouds become denser because a larger number of active and sleep windows is collected. Nevertheless, the qualitative behavior remains the same: the active phase stays in the high-power region, while the sleep phase settles at an intermediate level below active but above the fully gated sleep level.

The `idle before` and `idle after` groups are included as reference operating points. They should not be interpreted as low-power states, because the processor is not in the clock-gated sleep condition during these intervals. Their role is to verify that the operating point before and after the benchmark remains stable and that the reduction observed during sleep is related to the inserted `WFI` interval rather than to a global drift in the measurement.

The numerical results obtained from the phase-wise acquisition are summarized in Table 7.8. The active power remains close to 85.6 mW, while the complete sleep average is approximately 66.5 mW. When the initial transition is excluded, the steady sleep level is approximately 64.7 mW. Therefore, the active-to-sleep steady-state reduction is about 20.9 mW.

Table 7.8: Arty A7-100T XADC results for the Sleep-aware configuration (SLEEP).

Iterations	P_{act} (mW)	σ_{act} (mW)	$P_{\text{slp,tot}}$ (mW)	$\sigma_{\text{slp,tot}}$ (mW)	$P_{\text{slp,ss}}$ (mW)	$\sigma_{\text{slp,ss}}$ (mW)	$\Delta P_{\text{act}-\text{slp,ss}}$ (mW)
1	85.61	2.24	66.49	5.84	64.70	1.66	20.91
2	85.64	2.03	66.54	5.90	64.72	2.06	20.92
5	85.53	2.10	66.45	5.98	64.68	2.00	20.85
10	85.71	2.06	66.68	5.96	64.86	2.15	20.85
20	85.47	2.07	66.49	5.93	64.70	2.00	20.77
50	85.40	2.06	66.40	6.01	64.58	2.02	20.82

Table 7.8 shows that the Sleep-aware configuration (SLEEP) produces a repeatable reduction between the active phase and the WFI-based sleep interval. The active power remains close to 85.5 mW for all iteration counts, while the complete sleep-window average remains around 66.5 mW. When the initial sleep transition is excluded, the steady sleep level is approximately 64.6–64.9 mW.

The resulting active-to-sleep steady-state difference remains close to 21 mW across the tested iteration range. This confirms that software-level sleeping through `WFI` reduces the measured activity with respect to active execution. However, this reduction is still smaller than the one obtained in the fully clock-gated configuration, where the sleep level drops to approximately 50 mW. Therefore, the SLEEP case confirms that `WFI` alone is beneficial, but it does not suppress the residual clock-driven switching activity as effectively as the BUFGCE-based gated implementation.

Figure 7.27 compares the average active, sleep-total, and sleep-steady power values across the tested iteration counts. The active phase remains around 85 mW, whereas the sleep steady-state level remains around 65 mW. This confirms that the software sleep

mechanism produces a measurable activity reduction, but not as large as in the fully clock-gated implementation.

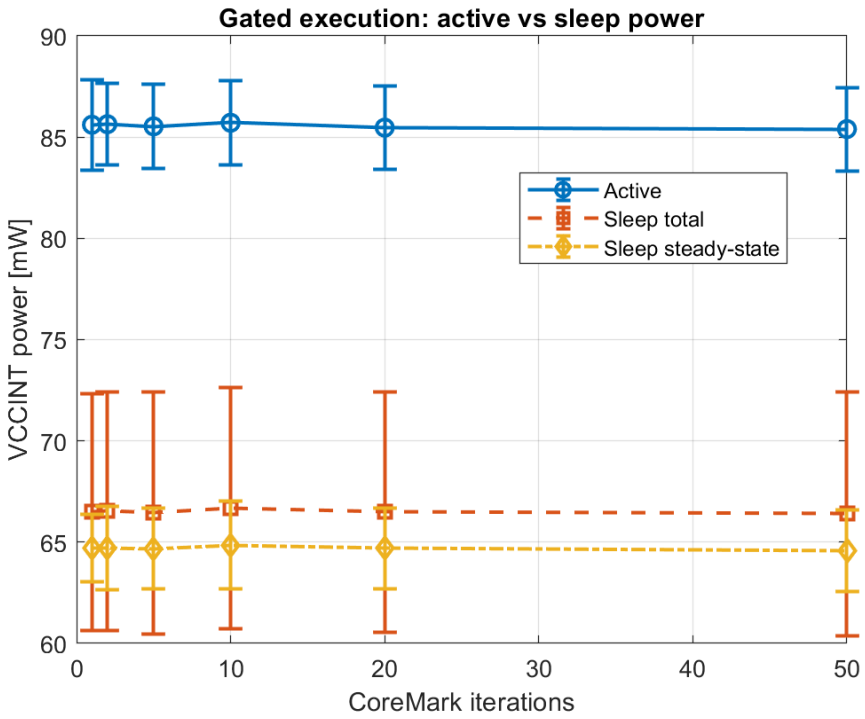


Figure 7.27: Average active, sleep-total, and sleep-steady VCCINT power for the Sleep-aware configuration (SLEEP). The sleep steady-state level remains significantly higher than in the fully clock-gated configuration.

The corresponding power differences are shown in Fig. 7.28. The difference between active and sleep-total remains around 19 mW, while the difference between active and sleep-steady is around 21 mW. The slightly larger steady-state difference is expected because the first sleep samples still include the transition from the active level.

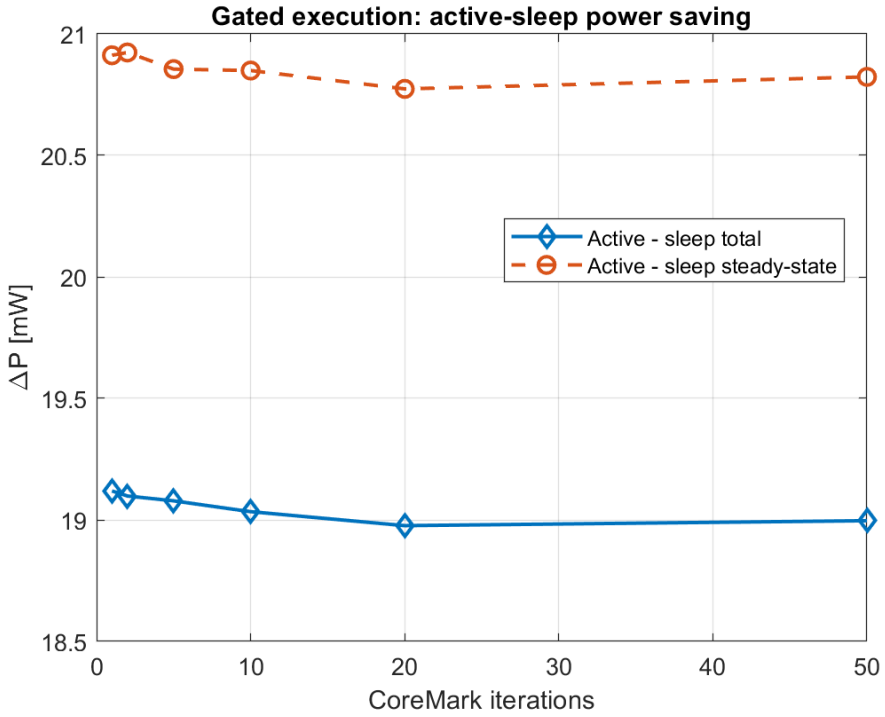


Figure 7.28: Active-sleep power difference for the Sleep-aware configuration (SLEEP). The steady-state metric excludes the initial transition and therefore provides a slightly larger difference than the complete sleep-window average.

The comparison with the idle reference is reported in Fig. 7.29. This figure is not used as the main metric for power saving, because the idle-before and idle-after windows do not correspond to a true low-power state. However, it confirms that the idle reference remains close to the active operating region, supporting the interpretation that the sleep reduction is caused by the explicit `WFI`-based interval.

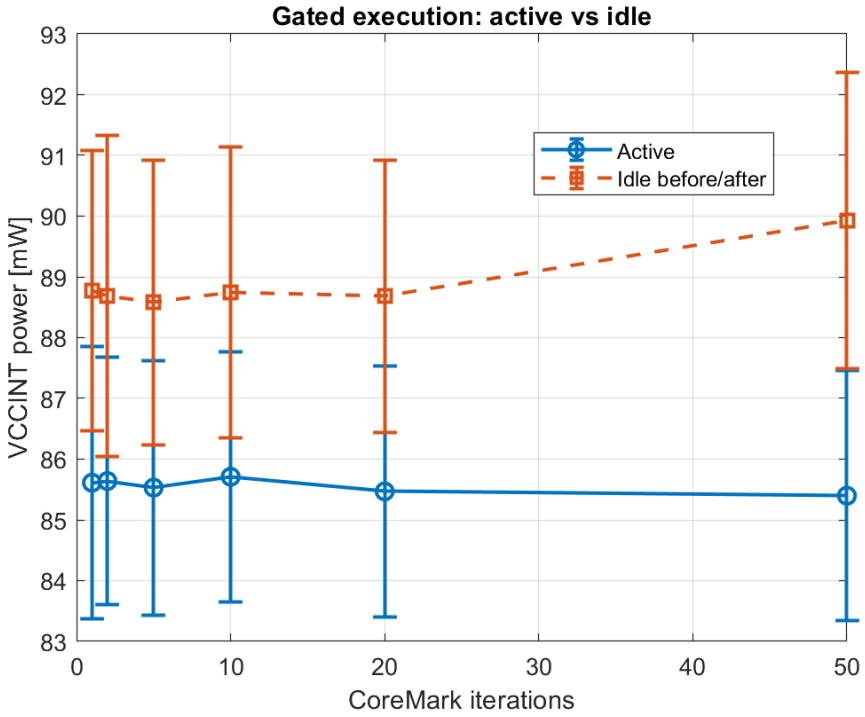


Figure 7.29: Active and idle-reference VCCINT power for the Sleep-aware configuration (SLEEP). The idle-before and idle-after samples are used only as reference operating points and not as low-power sleep measurements.

Overall, the Sleep-aware configuration (SLEEP) confirms that software-level sleeping through `WFI` reduces the measured activity with respect to active execution. However, the sleep steady-state level remains substantially higher than in the fully clock-gated configuration. This demonstrates that entering a sleep state in software is not sufficient to obtain the maximum reduction in dynamic activity: the additional benefit is obtained only when the CPU clock is physically suppressed by the BUFGCE-based gating mechanism.

7.2.5 CE1 Control Configuration (CE1)

The CE = 1 control configuration was introduced to verify whether the reduction observed in the clock-gated implementation is caused by actual clock suppression or only by the presence of the modified BUFGCE-based clock path. In this configuration, the BUFGCE primitive remains instantiated in the design, but its enable input is forced permanently high:

$$\texttt{ce\_cpu} = 1.$$

For consistency with the CGATE analysis, the CE1 low-activity interval was processed using the same two metrics. The complete low window includes all samples acquired during the software-defined idle interval, while the steady-state low value excludes all samples with phase-local index lower than 75. This ensures that the CE1 control case and the clock-gated case are compared using the same treatment of the initial active-to-low transition. As a consequence, the CPU clock is always propagated. The software still executes the same chunked CoreMark structure with `WFI`-based idle intervals, but the clock is not physically disabled during those intervals. Therefore, this configuration

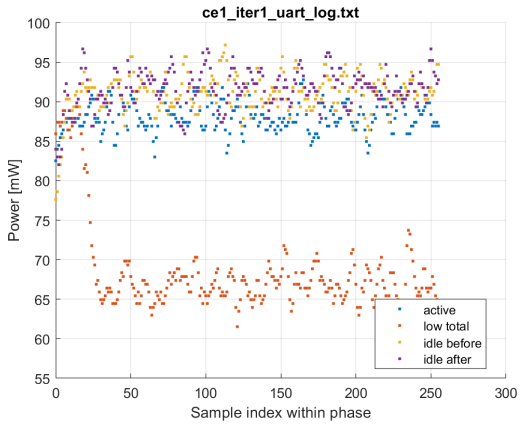
acts as a control case: it preserves the modified clock path and the same software timing structure, while removing the actual clock-gating action.

In the phase-wise XADC acquisition, four groups are represented:

- `active post-chunk`: samples acquired during the computation phases;
- `idle between chunks`: samples acquired during the `WFI`-based idle intervals between CoreMark chunks;
- `idle before`: reference samples acquired before benchmark execution;
- `idle after`: reference samples acquired after benchmark execution.

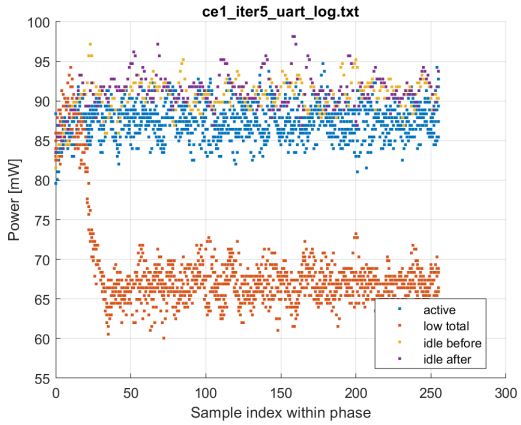
The most relevant comparison in this configuration is between the `active post-chunk` and `idle between chunks` groups. The `idle before` and `idle after` groups are kept as reference operating points and should not be interpreted as low-power states. They represent the system condition before and after the benchmark sequence and are used to check the stability of the acquisition.

Figure 7.30 reports the phase-wise XADC samples for the CE = 1 configuration over different CoreMark iteration counts. The active samples remain clustered in the upper region, around 85–90 mW. The samples acquired during the idle intervals between chunks are lower, around 65–70 mW, showing that the `WFI` instruction reduces useful switching activity. However, this level remains well above the lower-power region observed in the fully clock-gated configuration, confirming that clock-driven activity is still present when `ce_cpu` is forced to one.

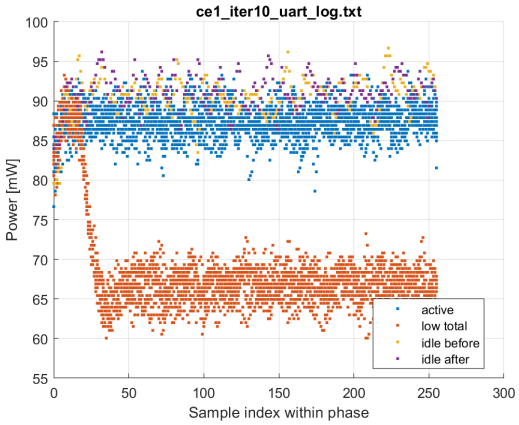


(a) 1 CoreMark iteration.

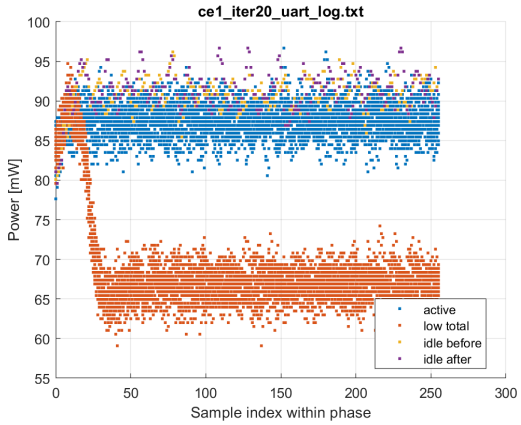
(b) 2 CoreMark iterations.



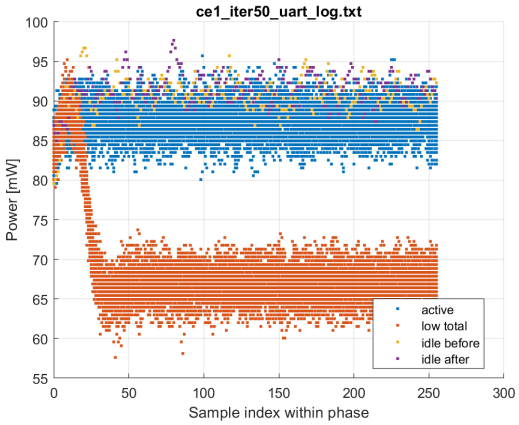
(c) 5 CoreMark iterations.



(d) 10 CoreMark iterations.



(e) 20 CoreMark iterations.



(f) 50 CoreMark iterations.

Figure 7.30: Phase-wise XADC power samples for the CE = 1 control configuration on the Arty A7-100T platform. The clock-gating path is present, but the BUFGCE enable is forced permanently high. The `idle between chunks` samples are lower than the active samples because the processor executes `WFI`, but the clock is still physically propagated.

The six panels in Fig. 7.30 show that the CE = 1 behavior is repeatable across different benchmark lengths. As the number of iterations increases, the number of active and idle-between-chunks samples also increases, producing denser point clouds. Nevertheless, the

separation between the active region and the idle-between-chunks region remains visible. This confirms that the `WFI`-based idle intervals reduce the measured activity even when the clock remains enabled.

However, the idle-between-chunks level does not reach the lower values observed in the fully clock-gated configuration. This is the key distinction: in CE = 1 the processor is logically idle, but the clock network and the clocked sequential elements are still active. Therefore, this configuration quantifies the effect of software-level sleeping without effective hardware clock suppression.

The numerical results extracted from the CE = 1 measurements are summarized in Table 7.9. The active power remains approximately constant around 87.5–88.1 mW, while the idle-between-chunks power remains around 68.4–68.6 mW. The corresponding active-to-idle-between difference is approximately 19 mW over the complete iteration range.

Table 7.9: Arty A7-100T XADC results for the CE1 control configuration. The complete low window includes the initial active-to-low transition, while the steady-state low value excludes samples with phase-local index lower than 75.

Iterations	P_{active} (mW)	σ_{active} (mW)	$P_{\text{low,total}}$ (mW)	$\sigma_{\text{low,total}}$ (mW)	$P_{\text{low,steady}}$ (mW)	$\sigma_{\text{low,steady}}$ (mW)
1	88.08	1.89	68.61	5.94	66.99	2.01
2	87.54	2.24	68.52	6.00	66.75	2.06
5	87.51	2.20	68.61	5.95	66.89	2.00
10	87.47	2.06	68.38	6.00	66.63	1.99
20	87.61	2.08	68.56	5.94	66.79	1.92
50	87.66	2.10	68.62	5.97	66.83	2.00

The CE1 results confirm that the modified BUFGCE clock path alone does not produce the low-power behavior observed in the clock-gated configuration. The active power remains close to 87.5 mW, while the complete low-window power remains close to 68.5 mW. After excluding the initial active-to-low transition, the steady-state low value settles around 66.8 mW.

As in the clock-gated case, the standard deviation of the complete low window is higher than the steady-state value because the complete window includes the transition after entering the software-defined idle interval. After excluding the first 75 phase-local samples, the standard deviation decreases from approximately 6 mW to about 2 mW. However, the steady-state CE1 low level remains significantly higher than the corresponding clock-gated value, confirming that the CPU clock is still physically propagated in this control configuration.

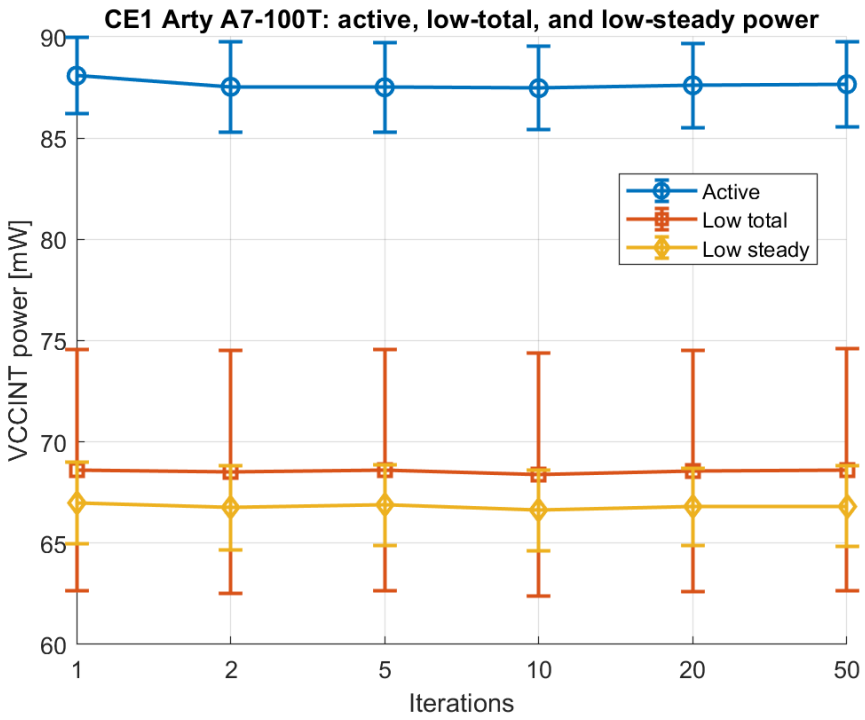


Figure 7.31: Average active, complete low-window, and steady-state low-window V_{CCINT} power for the CE1 control configuration. Error bars represent the standard deviation of the XADC samples acquired in each phase. The steady-state low values are computed after excluding all low-phase samples with phase-local index lower than 75. Lines are drawn only as visual guides between the tested iteration counts.

Figure 7.31 summarizes the reconstructed V_{CCINT} power for the active, complete low-window, and steady-state low-window regions of the CE1 control configuration. The active power remains close to 87.5 mW, while the complete low-window power remains around 68.5 mW. After excluding the initial active-to-low transition, the steady-state low-window value settles around 66.8 mW.

The reduction from active to low activity confirms that the WFI-based software idle interval lowers the measured activity even when the BUFGCE clock path is permanently enabled. However, the CE1 low-power level remains significantly higher than the corresponding clock-gated value, confirming that software sleep alone does not provide the same reduction as actual CPU clock suppression.

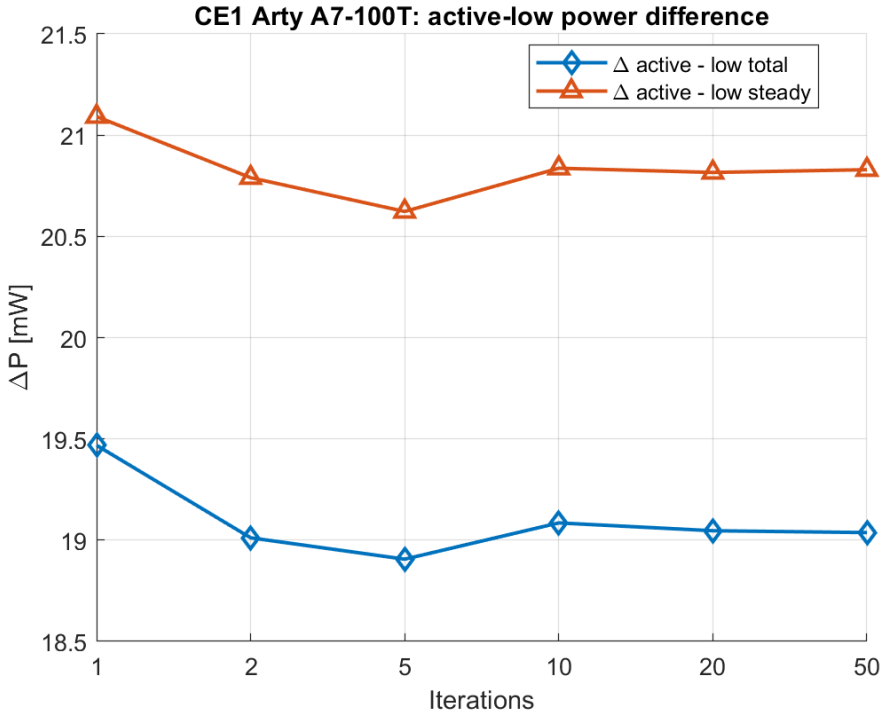


Figure 7.32: Measured active-low power difference for the CE1 control configuration, computed both over the complete low-activity window and over the steady-state low-activity region. Lines are drawn only as visual guides between the tested iteration counts.

Figure 7.32 reports the active-low power difference for the CE1 control configuration. When the complete low-activity window is considered, the difference remains close to 19 mW. When the initial transition samples are excluded, the active-to-steady-low difference increases to approximately 20.8 mW.

This confirms that the software-defined idle interval reduces the activity with respect to active execution even when the BUFGCE clock path is permanently enabled. However, the resulting low-activity level remains significantly higher than in the clock-gated configuration, showing that WFI-based idle behavior alone does not provide the same benefit as actual CPU clock suppression.

7.2.6 Comparison of the Arty A7-100T Configurations

The Arty A7-100T measurements make it possible to separate the effects of software sleep and hardware clock gating. The continuous baseline establishes the idle reference level of the platform. The SLEEP and CE = 1 configurations show the effect of entering `WFI` without effective clock suppression. Finally, the fully clock-gated configuration shows the additional reduction obtained when the CPU clock is disabled during the sleep interval.

For this comparison, three power levels are considered. The active level corresponds to the execution of CoreMark chunks. The low-total level corresponds to the complete low-activity interval, including the initial transition after `WFI`. The low-steady level corresponds to the settled region after the initial transition. In the baseline case, no active/sleep separation exists, so only the idle reference before and after benchmark execution is reported.

For the phase-aware configurations, namely SLEEP, CE1, and CGATE, the low-total

value is computed over the complete software-defined low-activity interval. The low-steady value is computed by excluding all low-phase samples with phase-local index lower than 75. This separates the initial active-to-low transition from the settled low-activity region and applies the same post-processing rule to all configurations.

Table 7.10: Comparison of the main Arty A7-100T XADC measurement results.

Configuration	P_{active} (mW)	$P_{\text{low,total}}$ (mW)	$P_{\text{low,steady}}$ (mW)	Interpretation
Continuous baseline	–	–	$\sim$ 73–75	Stable idle reference before/after execution
SLEEP / WFI-only	$\sim$ 85.5	$\sim$ 66.5	$\sim$ 64.7	Software sleep reduces activity, but the clock remains active
CE1 control	$\sim$ 87.6	$\sim$ 68.5	$\sim$ 66.8	BUFGCE path present, clock permanently enabled
CGATE	$\sim$ 85.3	$\sim$ 50.6	$\sim$ 47.2	WFI plus effective CPU clock suppression

Table 7.10 shows that the clock-gated sleep level is substantially lower than both WFI-only references. This confirms that the observed reduction is not only due to the processor stopping useful instruction execution, but also to the suppression of the clock signal in the CPU clock domain. The continuous baseline is not directly comparable in terms of active/sleep separation, because no explicit sleep window exists in that configuration; it is therefore used only as an idle reference.

A more direct quantification of the clock-gating contribution is obtained by comparing the CE = 1 control configuration with the clock-gated implementation. These two cases share the same software structure and the same BUFGCE-based clock path, but only the clock-gated configuration actually disables the CPU clock during the low-activity interval.

Table 7.11: Comparison of complete and steady-state low-activity V_{CCINT} power between the CE1 control configuration and the clock-gated configuration.

Iters	$P_{\text{CE1,total}}$ (mW)	$P_{\text{CGATE,total}}$ (mW)	ΔP_{total} (mW)	Saving (%)	$P_{\text{CE1,steady}}$ (mW)	$P_{\text{CGATE,steady}}$ (mW)	ΔP_{steady} (mW)	Saving (%)
1	68.61	50.71	17.90	26.09	66.99	47.33	19.66	29.35
2	68.52	50.74	17.78	25.95	66.75	47.45	19.30	28.91
5	68.60	50.59	18.01	26.25	66.89	47.28	19.61	29.32
10	68.38	50.54	17.83	26.08	66.63	47.17	19.46	29.21
20	68.56	50.36	18.20	26.55	66.79	46.98	19.81	29.66
50	68.62	50.58	18.04	26.29	66.83	47.14	19.70	29.47

The complete-window saving is computed as

$$\text{Saving}_{\text{total}} = \frac{P_{\text{CE1,total}} - P_{\text{CGATE,total}}}{P_{\text{CE1,total}}} \cdot 100.$$

The steady-state saving is computed analogously after excluding the initial transition samples from both configurations:

$$\text{Saving}_{\text{steady}} = \frac{P_{\text{CE1,steady}} - P_{\text{CGATE,steady}}}{P_{\text{CE1,steady}}} \cdot 100.$$

Using the complete low-activity window, the clock-gated configuration reduces the measured V_{CCINT} power by approximately 18 mW with respect to the CE1 control case, corresponding to a saving close to 26%. When the initial active-to-low transition samples are excluded from both configurations, the steady-state difference increases to approximately 19.5–20 mW, corresponding to a saving close to 29%.

Both comparisons lead to the same conclusion: the lower low-activity power observed in the clock-gated configuration is not caused only by the execution of `WFI`, but by the actual suppression of the CPU clock during the low-activity interval.

Figure 7.33 compares the active-to-low steady-state power difference for the phase-aware configurations. The SLEEP and CE1 cases remain around the 20 mW range, whereas the fully clock-gated configuration reaches approximately 38 mW. This shows that `WFI` alone provides a measurable reduction, but the largest separation between active execution and low-activity intervals is achieved only when the CPU clock is physically suppressed.

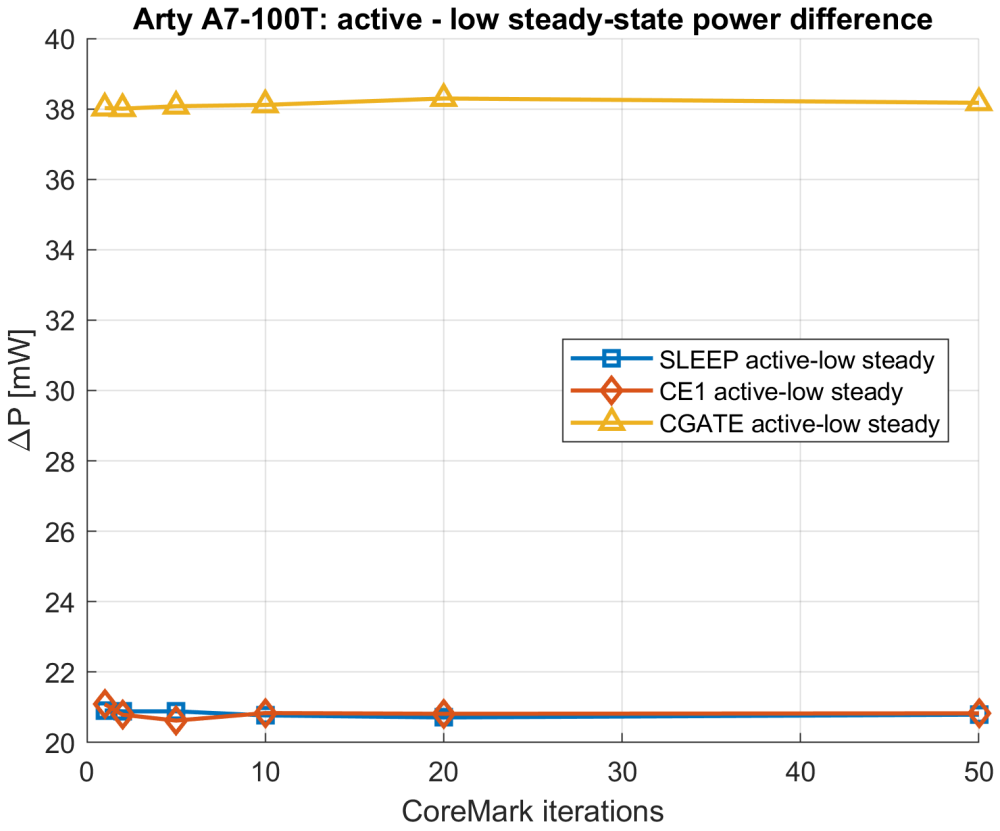


Figure 7.33: Active-to-low steady-state power difference for the Arty A7-100T phase-aware configurations. The clock-gated configuration shows the largest separation between active execution and the settled low-activity interval, confirming the additional contribution of hardware clock suppression. Lines are drawn only as visual guides between the tested iteration counts.

The same conclusion can be observed from the bar comparison at 50 iterations, shown in Fig. 7.34. The continuous baseline provides only an idle reference, since no explicit active/sleep phase separation is present. The SLEEP and CE = 1 configurations show an intermediate low-power level, because the processor enters `WFI` but the clock remains active. The clock-gated configuration exhibits the lowest low-total and low-steady values, demonstrating that the BUFGCE-based suppression of the CPU clock provides the additional reduction.

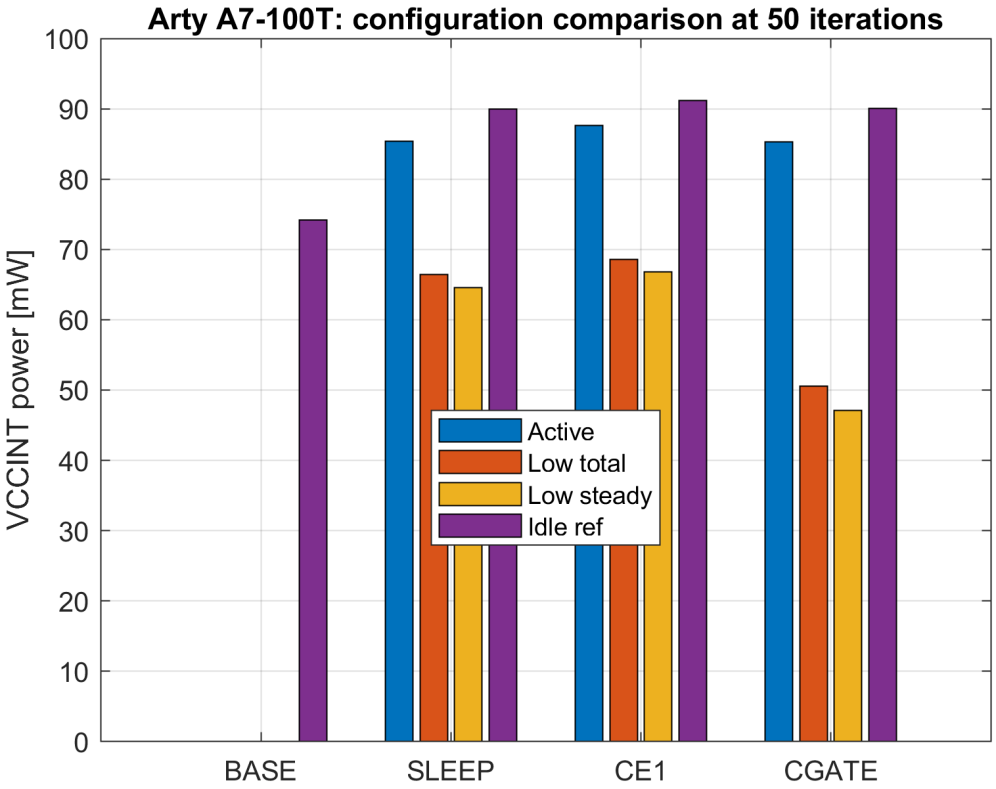


Figure 7.34: Comparison of active, low-total, low-steady, and idle-reference V_{CCINT} power at 50 CoreMark iterations. The clock-gated configuration exhibits the lowest low-activity level, while the CE1 and SLEEP configurations remain at an intermediate level because the CPU clock is still propagated.

The comparison can be interpreted in two steps. First, the transition from active execution to `WFI` without effective clock gating reduces the measured power by approximately 19–21 mW. This represents the contribution of software-level sleep and the reduction of useful switching activity. Second, comparing CE1 and CGATE isolates the contribution of actual clock suppression. This contribution is approximately 18 mW when the complete low window is considered and approximately 19.5–20 mW in the steady-state low region.

The remaining power in the gated sleep phase is expected and physically meaningful. The gated clock domain does not include the entire FPGA. The XADC monitor, the UART interface, the CLINT timer, the global clocking resources, leakage power, and other non-gated logic remain active. Therefore, the sleep power cannot drop to zero. The important result is the relative reduction with respect to the CE = 1 and WFI-only configurations.

Overall, the Arty A7-100T campaign confirms the same conclusion obtained from the previous CW305 measurements: software-defined sleep intervals alone reduce useful switching activity, but the largest and most reliable reduction is obtained only when the sleep state is combined with actual hardware clock suppression. The phase-aware FIFO and capture-enable mechanism make this conclusion more robust, because each XADC sample is associated with the hardware phase in which it was acquired.

Chapter 8

Conclusion

8.1 Summary of the Work

This thesis investigated whether a standard RISC-V sleep mechanism can be combined with FPGA-safe hardware clock gating to reduce unnecessary dynamic activity in an FPGA-based RISC-V controller. The target architecture was the NEORV32 processor, implemented on Xilinx Artix-7 FPGA platforms, and the experimental workload was a modified CoreMark benchmark with controlled active and sleep intervals.

The main question addressed by the work was whether software-defined idle phases are sufficient to reduce power-related activity, or whether actual hardware clock suppression is required. The answer obtained from the experimental campaign is clear: executing WFI reduces useful processor activity, but it does not by itself eliminate the clock-driven switching that remains when the CPU clock is still physically propagated. The largest reduction is obtained only when the WFI-based sleep state is connected to a real clock-gating mechanism.

To evaluate this point, the NEORV32 system was modified at both hardware and software level. On the hardware side, a BUFGCE primitive was used to generate a dedicated CPU clock domain, `clk_cpu`. The corresponding enable condition was derived from the processor sleep state and from the required wake-up sources, including timer and interrupt signals. On the software side, CoreMark was divided into computation chunks separated by timer-driven WFI intervals. This made it possible to compare continuous execution, WFI-only sleep, a permanently enabled clock-path control case, and the fully clock-gated implementation.

The validation combined implementation-level analysis and physical measurements. The CW305 platform was used for oscilloscope-based observation of the trigger and shunt-related activity waveform, while the Arty A7-100T platform was used for an integrated XADC-based measurement flow. On the Arty platform, the acquisition system was improved by storing each XADC sample together with the phase code active at the time of acquisition, avoiding software-side phase mislabeling during UART readout.

8.2 Main Contributions and Experimental Findings

The first contribution of this thesis is the implementation of a standard-compliant low-power mechanism for an FPGA-based RISC-V soft processor. Instead of introducing a custom sleep instruction or a non-standard software control register, the proposed approach uses the RISC-V WFI instruction as the software-visible entry point into the

low-activity state. This makes the mechanism conceptually portable to other RISC-V systems, provided that the hardware clock-control primitive is adapted to the target FPGA technology.

The second contribution is the experimental separation between software sleep, clock-path modification, and actual hardware clock suppression. The SLEEP configuration isolates the effect of WFI-based idle intervals without effective clock gating. The CE1 configuration keeps the BUFGCE path present but forces the clock enable permanently high, making it possible to distinguish the effect of the modified clock path from the effect of disabling the clock. The CGATE configuration combines the same WFI-based idle intervals with dynamic BUFGCE-based CPU clock suppression.

The experimental results show that these effects are not equivalent. In the Arty A7-100T measurements, the SLEEP configuration reduced the measured activity with respect to active execution, but the low-activity level remained significantly higher than in the clock-gated case. This confirms that WFI alone reduces instruction execution activity, but does not remove the residual switching caused by the still-propagated CPU clock.

The CE1 control case further confirmed this interpretation. Since CE1 includes the same BUFGCE-based clock path but keeps the CPU clock permanently enabled, the difference between CE1 and CGATE can be attributed to effective clock suppression rather than to the mere insertion of the BUFGCE primitive. On the Arty A7-100T platform, the fully clock-gated configuration reduced the low-activity VCCINT power by approximately 18 mW with respect to CE1, corresponding to a reduction of about 26%. The active-to-sleep power difference in the clock-gated configuration remained close to 35 mW across the tested CoreMark iteration counts.

A further contribution is methodological. The phase-aware XADC acquisition flow made the Arty measurements more robust than a purely software-labelled sampling approach. By storing the phase code together with the raw XADC value at acquisition time, the measurement system avoided assigning incorrect phase labels to samples already present in the FIFO during UART dumping. This strengthened the interpretation of the active, sleep, and reference measurements.

8.3 Relation to Related Work and Practical Integration

The results are consistent with the general conclusion of previous FPGA clock-gating studies: reducing clock propagation can lower dynamic activity in FPGA designs. However, the contribution of this thesis is not only the observation of a power reduction. The key point is the controlled separation between WFI-only behavior, clock-path control, and actual hardware clock gating on a complete RISC-V soft-core processor.

Compared with related work based mainly on tool-level power estimation or on isolated functional blocks, this thesis provides a physical validation flow using two FPGA platforms. The CW305 setup gives direct oscilloscope-based evidence of activity modulation, while the Arty A7-100T setup provides a more automated phase-wise measurement flow. Therefore, the results complement simulation-based and implementation-level estimates with physical measurements of the implemented design.

In a real FPGA-based RISC-V controller, the proposed mechanism could be integrated as a low-power idle mode. The software would enter idle intervals through WFI, while the hardware would translate the processor sleep state into a safe clock-enable condition for the CPU-side clock domain. Critical infrastructure, such as timer logic, interrupt

handling, communication interfaces, debug access, and measurement or monitoring logic, must remain clocked so that the system can wake up correctly and remain observable.

The approach is therefore suitable for interrupt-driven embedded systems in which the processor alternates between computation windows and waiting intervals. UAV-like controllers are one example of this execution model, but the same principle applies to sensor-driven, event-driven, and periodically scheduled embedded workloads. The method does not require changing the nominal supply voltage or operating frequency, and it does not destroy processor state, unlike more invasive power-gating approaches.

8.4 Limitations

The results should be interpreted within the limits of the experimental setup. First, the workload is based on CoreMark. Although the benchmark was modified to reproduce controlled active and sleep intervals, it is not a complete UAV flight-control application. The results therefore validate the clock-gating mechanism under a repeatable benchmark workload, not the total energy saving of a real UAV controller.

Second, the measurements are platform- and measurement-chain-dependent. The CW305 oscilloscope-based measurements are useful for comparative analysis, but they are affected by oscilloscope resolution, bandwidth, probe loading, coupling mode, shunt tolerance, and environmental noise. For this reason, the CW305 results should be treated as activity-related comparative indicators rather than as absolute DC power measurements.

Third, the Arty A7-100T measurements are based on the board current-sense path and the FPGA XADC. This provides a repeatable and automated acquisition flow, but it is still limited by ADC resolution, sampling rate, analog front-end accuracy, and the UART logging procedure. The reconstructed VCCINT power values represent the monitored core rail and do not include the complete power consumption of the FPGA board, auxiliary rails, I/O banks, or external interfaces.

Fourth, the implemented gating strategy targets a selected CPU-related clock domain, not the complete SoC or the entire FPGA. The CLINT timer, UART interface, XADC monitor, global clocking resources, leakage power, and other non-gated logic remain active. This is necessary for correct wake-up and measurement operation, but it also means that the sleep power cannot drop to zero. The measured saving should therefore be understood as the reduction obtained by suppressing the CPU clock domain during WFI-based sleep intervals.

Finally, the implementation is experimentally validated on AMD/Xilinx Artix-7 platforms using the BUFGCE primitive. The architectural concept is not limited to AMD FPGAs, because WFI is a standard RISC-V mechanism. However, the clock-control primitive is technology-specific. Porting the method to Intel, Lattice, or other FPGA families would require mapping the design to the corresponding vendor-supported clock-control resource and repeating timing, wake-up, and power validation.

8.5 Future Work

Several research directions can extend this work.

A first direction is the evaluation of more realistic embedded workloads. A UAV-like test application including periodic sensor acquisition, control-loop computation, commu-

nication events, and actuator update phases would make the activity pattern closer to a real flight-control system. This would also allow the clock-gating strategy to be evaluated under realistic timing constraints and interrupt rates.

A second direction is a detailed wake-up latency characterization. This thesis focused mainly on functional correctness and power/activity reduction. Future work should measure the latency between the wake-up interrupt and the resumption of useful instruction execution, in order to quantify the timing cost introduced by the gated CPU clock domain.

A third direction is the extension of the technique to additional SoC domains. The present implementation gates the CPU-related clock domain, while several peripherals remain active. A more advanced design could introduce multiple independently gated clock domains, selectively disabling peripherals that are not required during specific idle intervals. This would require careful handling of clock-domain crossings, bus transactions, state retention, and wake-up dependencies.

A fourth direction is the combination of clock gating with other low-power techniques. Frequency scaling could reduce dynamic power during low-performance operating modes, while undervolting could further reduce power if timing margins are sufficient. Combining WFI-driven clock gating with dynamic frequency scaling or controlled undervolting could provide a broader power-management strategy. However, these techniques would require additional clock-generation support, voltage-control infrastructure, timing-margin analysis, and safety checks to avoid compromising reliable operation.

A fifth direction is the improvement of the measurement infrastructure. On-board monitoring could be enhanced with deeper hardware buffers, timestamped samples, DMA-based acquisition, or lower-overhead logging. External measurements could be improved through DC-coupled high-resolution current acquisition over multiple FPGA supply rails, allowing a more complete estimation of board-level power and energy per workload.

Finally, the method should be evaluated on other FPGA families. Since BUFGCE is an AMD/Xilinx-specific primitive, portability requires replacing it with the appropriate vendor-supported clock-control resource, such as Intel clock-control blocks or Lattice dynamic clock-control primitives. Such experiments would clarify which parts of the method are architecture-level and which parts are specific to the Artix-7 implementation.

8.6 Final Remarks

The central result of this thesis is that software sleep and hardware clock gating are not the same effect. WFI provides a standard RISC-V mechanism for entering idle intervals, but the main reduction in clock-driven dynamic activity appears only when this sleep state is connected to real hardware clock suppression.

The proposed BUFGCE-based implementation shows that this connection can be made in a practical FPGA-based RISC-V system while preserving wake-up capability and correct benchmark execution. The Arty A7-100T measurements, supported by the CW305 observations, show that CPU clock suppression during WFI-based sleep intervals is an effective low-power technique for NEORV32-based FPGA designs.

Although the work does not implement a complete UAV controller, it validates a useful building block for UAV-like and interrupt-driven embedded systems: reducing unnecessary processor activity during waiting intervals through standard RISC-V sleep semantics and safe FPGA clock-control primitives.

References

[1] B.-L. Tan, W.-K. Lee, K.-M. Mok, and H.-G. Goh, "Clock gating implementation on commercial field programmable gate array (FPGA)," in *2018 4th International Conference on Electrical, Electronics and System Engineering (ICEESE)*, Nov. 2018, pp. 102–106. DOI: 10.1109/ICEESE.2018.8703530. Accessed: Feb. 3, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/8703530/.

[2] M. H. Vo, "Hybrid data driven clock gating and data gating technique for better saving power in ALU RISC-v," Accessed: Feb. 3, 2026. [Online]. Available: https://ijeer.forexjournal.co.in/archive/volume-12/ijeer-120133.html.

[3] D. R. Sulaiman, "Using clock gating technique for energy reduction in portable computers," in *2008 International Conference on Computer and Communication Engineering*, May 2008, pp. 839–842. DOI: 10.1109/ICCCE.2008.4580723. Accessed: Feb. 3, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/4580723/.

[4] S. Kaxiras and M. Martonosi, *Computer Architecture Techniques for Power-Efficiency* (Synthesis Lectures on Computer Architecture). Cham: Springer International Publishing, 2008, ISBN: 978-3-031-00593-0 978-3-031-01721-6. DOI: 10.1007/978-3-031-01721-6. Accessed: May 29, 2026. [Online]. Available: https://link.springer.com/10.1007/978-3-031-01721-6.

[5] F. U. Ahmed, Z. T. Sandhie, M. U. Mohammed, A. H. B. Yousuf, and M. Chowdhury, "Energy efficient FDSOI and FinFET based power gating circuit using data retention transistor," in *2018 IEEE Nanotechnology Symposium (ANTS)*, Nov. 2018, pp. 1–4. DOI: 10.1109/NANOTECH.2018.8653556. Accessed: Feb. 19, 2026. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8653556.

[6] "Clock gating," ASIC-System on Chip-VLSI Design, Accessed: May 29, 2026. [Online]. Available: https://asic-soc.blogspot.com/2008/04/clock-gating.html.

[7] "7 series FPGAs clocking resources user guide (UG472)," 2018.

[8] "BUFGCE • vivado design suite 7 series FPGA and zynq 7000 SoC libraries guide (UG953) • reader • AMD technical information portal," Accessed: Feb. 6, 2026. [Online]. Available: https://docs.amd.com/r/en-US/ug953-vivado-7series-libraries/BUFGCE.

[9] O. Ogundairo and C. Anggreani, "Energy-aware RTL techniques for multicore RISC-v SoCs in battery-powered embedded systems,"

[10] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 3, pp. 415–420, Mar. 2000, ISSN: 1558-1268. DOI: 10.1109/81.841927. Accessed: Feb. 3, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/841927/.

[11] T. Chindhu S. and N. Shanmugasundaram, "Clock gating techniques: An overview," in *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*, Mar. 2018, pp. 217–221. DOI: 10.1109/ICEDSS.2018.8544281. Accessed: Feb. 3, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/8544281/.

[12] S. V and A. Deshpande, "Low power implementation of RISC v processor with fine grained clock gating/power gating," in *2025 9th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, ISSN: 2767-1097, Nov. 2025, pp. 1–6. DOI: 10.1109/CSITSS67709.2025.11295650. Accessed: Feb. 3, 2026. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/11295650.

[13] C. Zhao and Y. Wang, "Clock gating circuit design based on data-driven improvements," in *Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering*, ser. EITCE '21, New York, NY, USA: Association for Computing Machinery, Dec. 31, 2022, pp. 54–58, ISBN: 978-1-4503-8432-2. DOI: 10.1145/3501409.3501420. Accessed: Feb. 3, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3501409.3501420.

[14] E. Bezati, S. Casale-Brunet, M. Mattavelli, and J. W. Janneck, "Clock-gating of streaming applications for energy efficient implementations on FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 4, pp. 699–703, Apr. 2017, ISSN: 1937-4151. DOI: 10.1109/TCAD.2016.2597215. Accessed: Feb. 3, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/7529097/.

[15] E. Bezati, S. C. Brunet, M. Mattavelli, and J. W. Janneck, "Coarse grain clock gating of streaming applications in programmable logic implementations," in *Proceedings of the 2014 Electronic System Level Synthesis Conference (ESLsyn)*, May 2014, pp. 1–6. DOI: 10.1109/ESLsyn.2014.6850387. Accessed: Feb. 3, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/6850387/.

[16] "Vivado design suite user guide design flows overview," 2022.

[17] stnolting et al., *Stnolting/neorv32: V1.12.7*, version v1.12.7, Language: en, Jan. 12, 2026. DOI: 10.5281/ZENODO.5018888. Accessed: Feb. 3, 2026. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.5018888.

[18] T. Kojima, M. Morita, H. Takase, and H. Nakamura, "An open-source framework for efficient side-channel analysis on cryptographic implementations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2026, ISSN: 1937-4151. DOI: 10.1109/TCAD.2026.3654878. Accessed: Feb. 3, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/11354523/.

[19] A. Dewar, J.-P. Thibault, and C. O'Flynn, "NAEAN0010: Power analysis on FPGA implementation of AES using CW305 & ChipWhisperer©r,"

[20] "CW305 artix FPGA target — ChipWhisperer documentation," Accessed: Feb. 10, 2026. [Online]. Available: https://chipwhisperer.readthedocs.io/en/latest/Targets/CW305%20Artix%20FPGA.html.

[21] "Arty a7 reference manual - digilent reference," Accessed: Feb. 10, 2026. [Online]. Available: https://digilent.com/reference/programmable-logic/arty-a7/reference-manual?srsltid=AfmBOoqkpzsO4Gqz1_eL86AumVIO21Fy04bKnrp4F8QziKOfGXtakz1

[22] "Arty a7 - digilent reference," Accessed: Feb. 12, 2026. [Online]. Available: https://digilent.com/reference/programmable-logic/arty-a7/start?srsltid=AfmBOoqUjg6PUNEesmFCW9-6cB7tsAqhqIhtT1u_jQe_RYhCVjOliKwU.

[23] "GTKWave documentation," Accessed: Feb. 4, 2026. [Online]. Available: https://gtkwave.github.io/gtkwave/.

[24] "About GHDL — GHDL 1.0-dev documentation," Accessed: Feb. 12, 2026. [Online]. Available: https://ghdl-rad.readthedocs.io/en/latest/about.html.